Алексей Русаков

Полное затмение

В начале лета известная компания Macromedia объявила о своих планах присоединиться к Eclipse Foundation и разработке собственной версии IDE на базе Eclipse — Zorn. Список этой организации насчитывает около 100 членов, среди которых встречаются весьма громкие имена. Попробуем разобраться, чем же так привлекательна эта платформа.

Немного истории

Eclipse — это попытка создать некую обобщенную среду разработки. Проект зародился в недрах компании IBM, которая в 2001 году открыла его исходный код для широкой публики, одновременно создав организацию Eclipse.org. В разное время нее вступали Borland, QNX, Red Hat, Fujitsu, Oracle, SAP, Ericsson и многие другие компании. Постепенно проект становился все более независимым от IBM, а в прошлом году стал «самостоятельной» некоммерческой организацией (несмотря на то, что активным центром разработки Eclipse попрежнему остается IBM).

В основе Eclipse лежит так называемая платформа (Eclipse Platform). Сама по себе она практически бесполезна и предоставляет лишь необходимый каркас. Остальные проекты работают поверх нее и являются различными средствами разработки и дополнениями (плагины) к ним и самой платформе. Среди них выделяются два основных проекта, номера версий которых совпадают с номерами версий платформы: это средства разработки на Java (Java Development Tools, JDT) и специализированная среда для разработки программ под платформу Eclipse (Plugin Development Environment). Кроме этого имеются средства разработки на C/C++ (C/C++ Development Tools, CDT), редактор UML-диаграмм, построитель графических интерфейсов (для Java). В процессе создания находятся средства для веб-программирования (Web Tools) и разработки на Python, а также некоторые другие проекты. Помимо eclipse.org есть сайты http://eclipseplugincentral.com и http://eclipse-plugins.info, содержащие информацию о многочисленных плагинах для Eclipse.

Внешний вид

Основная часть кода Eclipse написана на Java. Пользовательский интерфейс построен на основе библиотеки SWT (Standard Widget Toolkit), сочетающей Java и C++. Стоит отметить, что это не самостоятельная графическая библиотека, а кросс-платформенная оболочка для графических библиотек. Под Linux SWT использует библиотеку Gtk+ ; внешний вид Eclipse при этом соответствует выбранной теме Gtk+. Использование C++ в критичных к производительности местах позволило избежать медлительности, которой обычно грешат программы, написанные на Java. Увы, объем занимаемой Eclipse оперативной памяти сопоставим с крупным Java-приложением: если у вас меньше 512 Мбайт, работать будет не слишком комфортно.

Интерфейс Eclipse простым не назовешь. В чем-то это следствие обобщенности платформы, в чем-то сказывается принад-

лежность к IBM. Немного облегчает жизнь хорошая документация к JDT, в том числе введение в работу с Eclipse. Есть документация и для средств разработки на C/C++, но она заметно хуже. К сожалению, и сам интерфейс Eclipse, и документация на русский язык не переведены. Желающие могут заняться переводом, но учтите, что объем строк для перевода только CDT – 600 кбайт, то есть примерно 400 страниц текста.

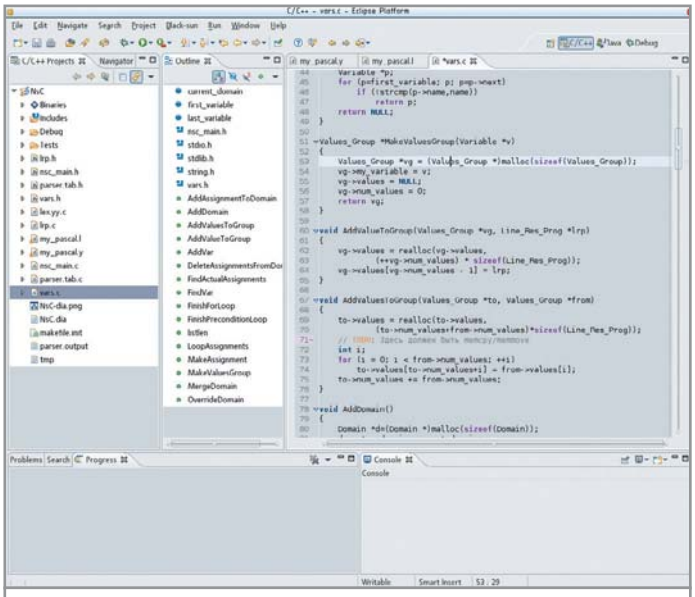
Eclipse – одна из самых функционально насыщенных сред разработки, как закрытых, так и открытых. По этой причине мы очень кратко остановимся на важных, на наш взгляд, моментах в работе с этой средой, а детальное исследование ее возможностей оставим читателю.

Для новичка самым непривычным в интерфейсе Eclipse будет, пожалуй, понятие перспективы (perspective). Она, в свою очередь, представляет собой набор и расположение панелей интерфейса – «видов» (views). Например, существуют отдельные перспективы для разработки на Java и C++, очень похожие друг на друга (вы можете вести разработку на двух языках в рамках одного проекта, по необходимости переключаясь между перспективами). Есть перспектива для отладки, в которой набор видов сильно отличается от аналогичного набора для разработки, а также перспектива для работы с системой управления версиями и другие.

Две перспективы могут не иметь абсолютно ничего общего между собой кроме открытых окон редактора. Открытые окна не меняются при смене перспективы. Иногда, как в случае переключения между разработкой и отладкой, это может быть весьма уместно. В иных случаях, например при переходе из C++ в Java, остающиеся открытыми окна с текстом на C++ могут раздражать.

Работа с проектами

Несмотря на обобщенность Eclipse, языков, на которых в этой среде можно нормально вести разработку, всего три: это Java, C и C++ (как вы уже поняли, Java поддерживается значительно лучше всех остальных). Мы постараемся побольше рассказать о C и C++, так как это самые распространенные языки в мире Open Source.



Написание реальной программы в Eclipse

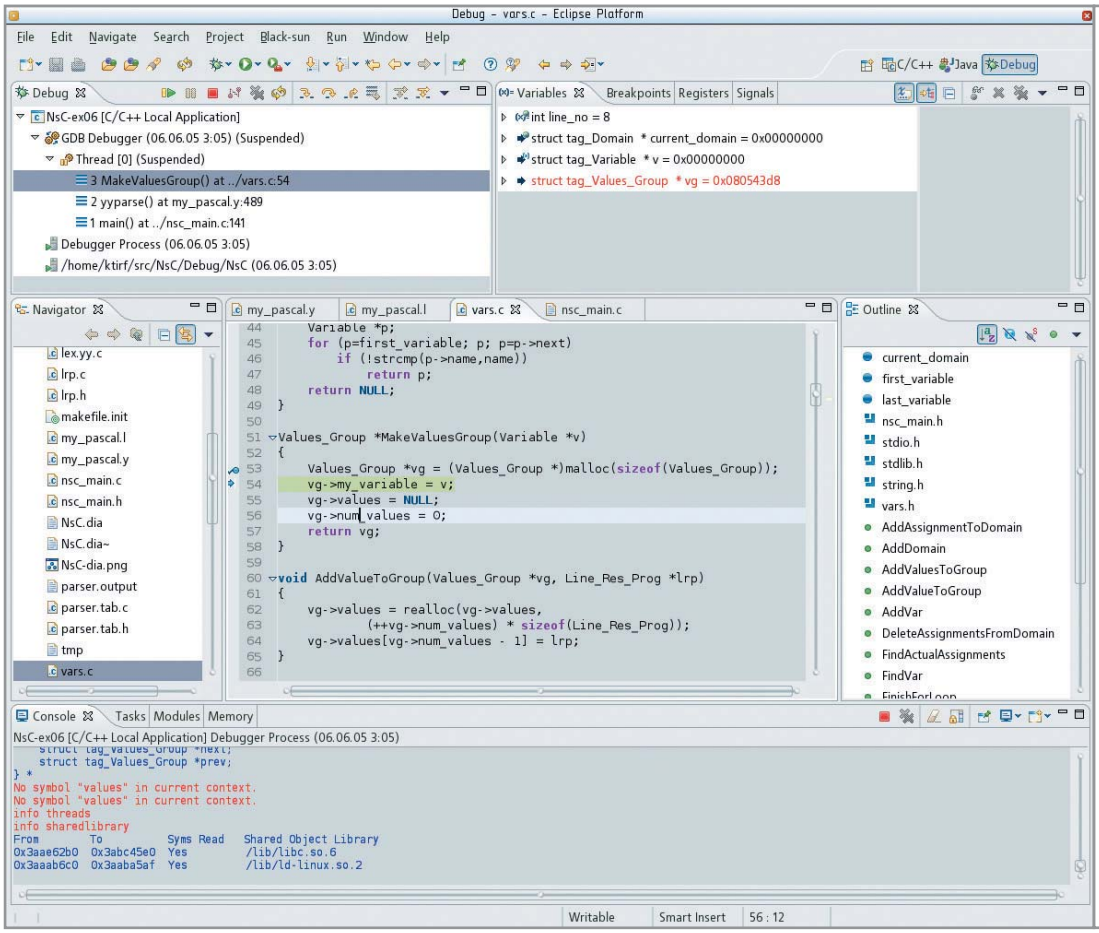
Поэтому начнем с CDT, то есть C/C++ Development Tools. Для C и C++ существует по два типа проектов – Standard Make и Managed Make. Если вы выбрали Standard Make, то Makefile будете писать сами, а среда лишь запустит Make, когда понадобится (вместо Make можно указать и другую программу). В проекте типа Managed Make среда берет на себя хлопоты по созданию Makefile, но и в этом случае остается возможность дописывания правил сборки вручную. Eclipse различает C и C++, особенно это касается проектов Managed Make, поэтому следите за тем, из какого языка вы выбираете тип проекта.

В CDT полностью отсутствуют средства создания графических интерфейсов. Можно использовать любимый Glade или QtDesigner, но интеграции построителя GUI в среду разработки, увы, нет никакой. Нет в Eclipse и явной поддержки GNU Autotools, хотя это, вобщем-то, и не мешает работать над использующими их проектами.

В остальном поддержка C и C++ реализована на уровне. Eclipse знает о шаблонах и пространствах имен C++, корректно работает с заголовочными файлами стандартной библиотеки, не имеющими расширения. Есть так называемые индексаторы (indexers), которые по своему назначению примерно соответствуют CTags (собственно, один из индексаторов в новой версии CDT как раз использует CTags). Элементы навигации показываются в двух списках — C/C++ Projects (с группировкой по проектам и файлам проектов) и Outline (только то, что в текущем файле). Немного неудобно то, что нет общего списка. С навигацией по проекту все в порядке; для быстрого перехода можно использовать как клавиатуру, так и контекстное меню, причем искать можно как объявления/определения, так и применения в тексте программы. В число элементов навигации входят и макросы.

Перейдем к Java Development Tools (JDT). Здесь всего один тип проекта, для сборки которого используется Ant. Редактировать правила Ant вручную, как и в проекте Managed Make из CDT, можно, но обычно не нужно. Построитель графического интерфейса под названием Visual Editor, привязанный к JDT, можно скачать с сайта обновлений и дополнений (update.eclipse.org) в виде плагина, после чего начать конструировать графические интерфейсы, использующие библиотеку SWT. В JDT вместо индексатора используются встроенные средства виртуальной машины, позволяющие получать списки пакетов/классов/членов. К началу июня в JDT уже появилась достаточно внятная поддержка обобщенных типов (generics) и некоторых других возможностей из J2SE 5.0.

В Eclipse есть отдельная перспектива, не зависящая от языка программирования, для работы с CVS. Для набирающей популярность системы управления версиями Subversion аналогичная перспектива имеется в составе плагина, который можно скачать с сайта разработчиков Subversion. Здесь поддерживается максимум возможностей, включая аннотирование исходников, сравнение различных редакций файлов, манипуляции с тегами и прочее. Создавая новые проекты,



Отладка и отлов ошибок

можно наполнять их кодом, выгружая исходные тексты с указанного CVS-репозитория.

Причем, для того чтобы знать, в каком состоянии относительно CVS находится тот или иной файл, в другую перспективу переключаться не нужно. В любой момент информация о CVS-статусе файла показывается в каком угодно навигаторе проектов в виде маленького значка в углу иконки самого файла. Точно таким же образом можно узнавать и о том, в каких файлах были ошибки и предупреждения при сборке или проходе индексатора.

Редактирование исходных текстов

Редакторы в CDT и JDT похожи, хотя избранность Java налицо и здесь. Возможно, редакторы – это одна из самых сильных частей Eclipse по сравнению с другими средами разработки. Кроме уже традиционных складок (folds), подсветки синтаксиса и автоотступов разработчики добавили две по-своему уникальные вещи. Первая из них – это пометка изменений рядом с номерами строк, причем, что действительно полезно, изменения могут вычисляться не только по сравнению с последним сохраненным текстом, но и с версией этого файла в CVS. Вторая же приятная особенность заключается в том, что рядом с вертикальной полосой прокрутки было выделено пустое место, поперечные полоски на котором указывают, в какой части файла индексатором и/или компилятором выдавались ошибки или предупреждения. Если щелкнуть мышью по поперечной полоске, курсор попадет непосредственно на проблемную строку.

Автоматического форматирования кода в CDT нет – только автоотступы. В JDT есть не только автоматическое форматирование, которое в версии 3.1 можно настроить вплоть до мелочей, но и весьма приличные возможности переработки кода. Да и в целом редактор JDT более проработан и содержит массу приятных и полезных опций, хотя, чтобы их использовать, нередко приходится запоминать многочисленные комбинации клавиш и копаться в огромном окне настроек.

Автодополнение и подсказки о параметрах (calltips) работают во всех трех языках. После автодополнения имени функции или метода автоматически добавятся скобки, между которыми будет помещен курсор. Еще более мощным средством помощи при наборе являются шаблоны кода. Используются они следующим образом. Например, всем программистам на C известно обычное употребление функции malloc:

```
type *var = (type *)malloc(n*sizeof(type));
```

При наличии подходящего шаблона эта строка получается так: набираем название шаблона (прямо в тексте), вызываем автодополнение (по умолчанию – «Ctrl+Space»), которое разворачивает название шаблона в сам шаблон строки и ставит курсор на будущее имя типа в начале строки. Затем набираем имя типа (причем это происходит одновременно в трех местах – взамен каждого из слов type), нажимаем «Tab», вводим имя переменной, еще раз «Tab», затем выражение, которое в нашем примере заменено на «n». Еще одно нажатие «Tab» переносит нас в конец строки, и можно писать текст дальше. При этом значения приведения типа и sizeof будут подставлены автоматически.

В Eclipse «из коробки» шаблона для malloc, правда, нет, зато есть шаблоны для основных синтаксических конструкций и кое-что на тему printf. Написать свои шаблоны проще простого, с их помощью можно быстро создавать функции и классы, не затрачивая время на перемещения и набор повторяющихся фрагментов.

В такой монументальной среде разработки как-то неловко отдельно говорить о подсветке синтаксиса. Однако с подсветкой синтаксиса в Eclipse все хорошо до тех пор, пока вы работаете с Java, C/C++ и файлами описания сборки (для Ant и Make). Подсветки даже самого обыкновенного HTML у вас не будет, если вы не поставите нужный плагин. Стабильные плагины, дающие подсветку синтаксиса для большого количества языков, пока отсутствуют. К счастью, для частных случаев наподобие тех же HTML/XML/CSS решение уже есть в виде плагина под названием Black-sun (его можно найти на сайте проекта). Этот плагин вдобавок улучшает подсветку синтаксиса для Makefile.

Еще одна слабая сторона редактора – это закладки (bookmarks). Несмотря на поддержку такой функции, пользоваться ею не слишком удобно. Эту проблему можно успешно решить, установив плагин Quickmarks.

Сборка и отладка

Начнем с того, что и в JDT, и в CDT можно найти синтаксические ошибки без запуска компиляции. Упрощенная проверка синтаксиса ведется индексатором, который в состоянии найти такие ошибки, как необъявленная переменная или неправильно написанное имя класса. JDT еще и поможет сразу исправить ошибку (например, объявить необъявленную переменную; для отсутствующего метода создать заглушку) посредством нажатия «Ctrl+1».

Для компиляции в CDT используется GCC; в проектах Standard Make можно задействовать и другой компилятор благодаря полной свободе в написании Makefile. В случае с JDT также можно выбрать виртуальную машину Java и, соответственно, компилятор.

Все ошибки и предупреждения компилятора собраны на отдельной панели, которая называется «Problems». Прямо из нее с помощью двойного клика мышью можно попасть в строку кода, на которую ссылается то или иное сообщение. Например, в JDT нажатие комбинации клавиш «Ctrl+1» работает в том случае, если для строки есть возможность автоматического исправления.

На заметку

Любопытный факт

Не совсем понятно, откуда появилось название Eclipse (в переводе с английского «затмение»). Но обратите внимание, что в eclipse.org никогда не входила и не входит компания Sun. Более того, у Sun есть своя среда разработки для Java, появившаяся еще до Eclipse. Но разработчиков, использующих Eclipse, больше. Похоже, IBM действительно удалось затмить солнце.

Что касается отладки, то здесь CDT вовсю использует возможности отладчика GDB. Фактически CDT – это графическая оболочка для GDB, имеющая довольно широкие возможности, хотя визуализации расположения структур в памяти наподобие той, что сделана в другой известной графической оболочке для GDB – DDD, здесь нет. Это немало, но и не так уж много, к тому же все-таки хотелось бы иметь профилировщик и отладчик памяти (например, в виде обертки для valgrind). У JDT есть свой отладчик, использующий ту же перспективу, что и отладчик CDT; таким образом, присутствует возможность вести отладку кода на обоих языках практически в одинаковом интерфейсе.

Интеграция со средой пользователя

Ни одна среда разработки не может содержать всех средств, которые могут понадобиться программисту. Eclipse позволяет непосредственно из среды разработки запускать внешние программы для обработки кода или выполнения других сопутствующих действий. В аргументах для внешних программ можно указывать различные переменные, связанные с текущим файлом, проектом или Eclipse в целом. Один из примеров уже упоминался выше: поскольку нативной поддержки Autotools в CDT не предусмотрено, можно указать все необходимое в виде внешних приложений, а Eclipse обновит нужные виды, когда приложения завершатся.

Еще одна форма интеграции в пользовательскую среду состоит в том, что, поскольку Eclipse не в состоянии открыть, например, файлы редактора диаграмм Dia, было бы неплохо, чтобы этот редактор запускался автоматически при попытке открыть подобные файлы. Для этого придется немного поработать вручную, а именно в разделе «General» в настройках вписать нужные маски файлов и пути к открывающим их программам. Получать MIME-типы из среды пользователя Eclipse пока не научилась, и есть опасение, что вряд ли научится – кроссплатформенность дает о себе знать.

Резюме

Все-таки можно говорить о том, что на Eclipse неизгладимый отпечаток наложило IBM-происхождение. Она, как почтенный мейнфрейм IBM S/370, велика в размерах, довольно сложна в освоении и необычайно эффективна для решения ряда задач. Eclipse не слишком дружит с популярными в Linux инструментами, такими как Autotools, Glade, valgrind. Основная ориентация на Java тоже не способствует популярности в стане Open Source. Вероятно, дело в том, что главная целевая аудитория Eclipse – разработчики крупного коммерческого софта, а не энтузиасты. Но тогда вообще неясно, почему нет оболочки для valgrind.

В любом случае, если вы собираетесь писать консольные программы или демоны на C/C++ и не боитесь сложностей освоения среды разработки, обязательно попробуйте Eclipse: средства управления проектами, редактор исходного кода и отладчик в CDT — просто превосходные! Если же вы пишете программы на Java, то и так наверняка давно знаете, что Eclipse – самая популярная и одна из лучших сред разработки для этого языка.