

Разработка СПО промышленного уровня на примере компилятора GCC

Д.М. Мельник¹, А.А. Белеванцев¹

¹Институт системного программирования РАН, {dm, abel}@ispras.ru

Аннотация — В докладе описывается история разработки компилятора GCC, единственного промышленного компилятора с открытым исходным кодом. Показывается эволюция модели разработки за более чем 20-летнюю историю компилятора и влияние этой модели на его функциональность.

Ключевые слова — компиляторы, свободное ПО, GCC

I. Введение

GNU Compiler Collection (GCC) – это набор компиляторов, созданный в рамках проекта GNU [1]. Компилятор GCC является свободным программным обеспечением и распространяется Free Software Foundation (FSF) под лицензиями GNU GPL и GNU LGPL. GCC используется как стандартный компилятор во всех свободных UNIX-подобных системах, а также в некоторых закрытых системах (IBM S/390, HP-UX, до недавнего времени Mac OS X).

В настоящий момент последней версией компилятора является версия 4.5.1, выпущенная в июле 2010 года. В неё включена поддержка языков Си (версии C99), Си++ (версии C++98 и частично C++0x), Фортран 95 (и частично Фортран 2003), Objective-C 1.0, Ада, Java. Официально поддерживаются 34 целевых архитектуры, многие для нескольких процессоров или десятков вариантов процессоров. Кроме того, поддержка некоторых языков и многих архитектур разрабатывалась третьими сторонами и не включена в поставку FSF.

GCC является оптимизирующим компилятором и включает реализацию более 150 оптимизационных трансформаций. Из важнейших реализованных недавно оптимизаций

стоит отметить инфраструктуру для проведения оптимизаций во время компоновки (Link Time Optimizations, LTO), впервые представленную в версии 4.5.0.

GCC является компилятором промышленного уровня с высоким качеством кода, компилирует сам себя (т.н. bootstrap) без каких-либо предупреждений компиляции, и содержит большой набор регрессионных тестов (десятки тысяч). В следующих главах будет описана схема разработки GCC, которая позволила добиться этого результата.

II. История разработки GCC

Изначально разработка GCC управлялась одним человеком, первое время – Ричардом Столлманом, основателем FSF, а потом – назначенным им архитектором проекта. Такая схема работала в конце 80-х-начале 90-х годов XX века, во время распространения CISC-процессоров. В середине 90-х с бумом RISC-процессоров для поддержания компилятора в конкурентном состоянии требовалась реализация современных оптимизаций. Кроме того, была необходимость портировать компилятор на большое количество новых архитектур.

Большие усилия на эти задачи тратились компанией Cygnus, первой из тех, кто построил бизнес-модель на поддержке свободного ПО. Цели Cygnus и открытого сообщества Linux, Фортран и других в получении конкурентного свободного компилятора расходились с целями FSF, которая стремилась прежде всего сохранить стабильность GCC. В течение нескольких лет компиляторной команде Cygnus не удавалось добавить жела-

емую функциональность в официальную версию GCC [2].

В 1997 году это привело к созданию проекта EGCS, ответвления GCC, в которое включили всю реализованную на тот момент функциональность по добавлению новых оптимизаций, языков (Фортран 77), библиотек (первая версия `libstdc++`, библиотеки языка Си++). В 1999 году оба проекта объединились, когда текущая версия EGCS была названа официально GCC версии 2.95. Модель разработки, сложившаяся в EGCS, стала со временем моделью, применяемой с некоторыми изменениями в GCC до настоящего времени.

III. Модель разработки GCC

Разработка компилятора была организована в три стадии, каждая из которых занимает около двух месяцев. На первой стадии в главную ветвь разработки вливаются крупные изменения (новые оптимизации, поддержка новых целевых архитектур, изменения инфраструктуры). На второй стадии крупные изменения не допускаются, однако возможны небольшие добавления новой функциональности; компилятор стабилизируется. На третьей стадии никаких новых разработок не допускается, проводятся только исправления найденных ошибок.

После того, как количество ошибок становится меньше 100, открывается новая ветвь разработки, из которой будет выполнен выпуск новой версии (релиз) компилятора. Главная ветвь разработки обновляет свой номер (например, с 3.2 на 3.3) и переходит в первую стадию. В дальнейшем к релизной ветви будут применяться только исправления т.н. регрессий (*regression*), т.е. тех ошибок, которых не было в предыдущих версиях компилятора. Первая версия, выпущенная из релизной ветви, будет иметь номер *x.y.0*, а дальнейшие версии, содержащие только исправления найденных ошибок – номера *x.y.1*, *x.y.2* и т.д.

В последние несколько лет эта модель упрощена, и вторая стадия разработки исключена, а первая удлинена. После первой стадии незамедлительно начинается третья, допускающая лишь исправление найденных ошибок. Ветвь для выпуска новой версии не создается, пока остаются ошибки наивысшего приоритета – критическая ошибка компилятора или неправильная компиляция кода на одной из первичных архитектур, наиболее важных с точки зрения их популярности и занимаемой GCC доли на рынке компиляторов для этих архитектур. В настоящий момент [3] таковыми являются архитектуры *x86* и *x86-64* с ОС *Linux* и *FreeBSD*, *MIPS*, *PowerPC*, *SPARC* и *ARM*.

Критерии выпуска новой версии различны для первичных, вторичных и остальных архитектур. Для первичных архитектур не должно наблюдаться ни одной регрессии по сравнению с предыдущими версиями компилятора, как на поставляемом наборе тестов, так и на пользовательских программах (тогда эти ошибки должны быть внесены в официальную базу данных ошибок). Для вторичных архитектур достаточно успешной сборки компилятором самого себя и прохождения значительного большинства тестов. При этом для релиза имеет значение только качество поддержки языков Си и Си++, как наиболее популярных среди пользователей GCC. Эта схема позволяет выпускать версию с новой функциональностью (которая нумеруется как *x.y.0*) примерно раз в год.

Разработка компилятора не контролируется централизованно. Тем не менее, все изменения проходят строгую техническую проверку разработчиками, ответственными за определенную часть компилятора (*maintainers*). Как правило, это наиболее опытные и хорошо известные сообществу разработчики, проектирующие и реализующие основную функциональность компилятора, а также имеющие право одобрять изменения других, рядовых разработчиков. Процесс разработки и проверки полностью

открыт, все изменения и обсуждения публикуются через списки рассылки по электронной почте.

Политическими и административными вопросами, назначением ответственных разработчиков занимается организационный комитет (GCC Steering Committee) – небольшая группа уважаемых членов сообщества (не обязательно активных программистов, но с большим опытом разработки GCC). Комитет практически никогда не вмешивается в технические вопросы, за исключением спорных случаев (ответственные разработчики не могут прийти к соглашению по поводу определенного изменения). Например, комитет обсуждал с FSF изменения лицензии части исходного кода GCC, необходимые для поддержки встраиваемых дополнений (plugins), но никак не влиял на техническую реализацию необходимой функциональности.

Для исключения конфликта интересов и создания независимых клонов GCC, схожих с EGCS, в состав комитета включены программисты, работающие на разные компании – Red Hat, IBM, CodeSourcery, Novell, а также независимые программисты. Выпускающие новые версии (release managers) – разработчики, определяющие границы стадий разработки, содержание новой версии, приоритеты ошибок в базе ошибок – также работают на разные компании (Red Hat, Novell, CodeSourcery).

Наконец, необходимо заметить, что ядро наиболее активных разработчиков, реализующих основную функциональность компилятора, невелико – около 30 человек. Все такие программисты, а также подавляющее большинство остальных участников сообщества работают на компании IT-индустрии, либо производящие процессоры или компьютерные системы (Intel, IBM, AMD, ARM), либо широко использующие GCC в своих дистрибутивах Linux или просто в работе (Red Hat, Novell, Google), либо зарабатывающие на поддержке свободного ПО

(CodeSourcery, Adacore). Разработчиков из университетской среды и энтузиастов немного из-за огромной сложности компилятора, и в основном они представлены теми, кто работает над поддержкой языка Фортран.

IV. Заключение

Модель разработки GCC – единственного промышленного компилятора с открытым исходным кодом – сочетает полную открытость, строгую техническую проверку всех изменений компилятора, четкий критерий выпуска новых версий и баланс интересов различных компаний, влияющих на процесс разработки. В этой модели удобно работать блестящей команде разработчиков GCC, трудам которых компилятор обязан своему успеху на рынке.

Литература

- [1] GNU Compiler Collection. <http://gcc.gnu.org>.
- [2] The Short History of GCC Development. http://www.softpanorama.org/People/Stallman/history_of_gcc_development.shtml.
- [3] GCC 4.5 Release Criteria. <http://gcc.gnu.org/gcc-4.5/criteria.html>.