Gem #104: Функция gprbuild и конфигурационные файлы - Часть 1

Автор: Johannes Kanig, AdaCore

Краткое содержание: Данный Gem - первый в серии из трех публикаций, нацеленных на описание конфигурационных возможностей gprbuild. В этой публикации будет пояснено, как настроить gprbuild для работы с пользовательским компилятором с помощью конфигурационных файлов.

Давайте начнём…

Gem # 65 представил gprbuild, инструмент для создания программ GNAT, который поддерживает управление процессом сборки проекта Ada с файлом проекта. В частности, инструмент (утилита) gprbuild способен создавать проекты, которые используют исходные файлы в языках программирования, отличных от Ada, тогда как утилита gnatmake поддерживает только чистые проекты Ada.

Для Gprbuild нужны два файла: файл проекта (с расширением .gpr), определяющий характеристики проекта, такие как языки программирования, исходные каталоги, ключи компилятора, основные файлы и т. д. И файл конфигурации (с расширение .cgpr), описывающее используемые компиляторы.

Файлы проекта теперь широко используются, большинство инструментов GNAT знают, как обращаться с файлами проекта и как использовать информацию, которую они содержат, и многие проекты Ada теперь описываются с использованием файлов проекта. С другой стороны, файлы конфигурации не так просты в разработке. К счастью, инструмент gprconfig, который распространяется вместе с gprbuild, может автоматически генерировать файл конфигурации, который подходит для наиболее распространённых ситуаций. Ещё лучше, когда файл gprbuild (с использованием ключа -config) не передаётся конфигурационному файлу, gprbuild автоматически вызовет gprconfig.

В этом Gem #104 мы объясняем, как настроить gprbuild для использования настраиваемого компилятора.

В файле конфигурации перечислены компиляторы, которые будут использоваться для каждого языка, а также описываются параметры конфигурации, такие как необходимые ключи компилятора, суффикс для сгенерированных объектных файлов, переключатель командной строки для получения информации о зависимостях и т. д. Давайте посмотрим на простой пример. Хотя определение собственного компилятора полезно в основном для нестандартных компиляторов, в нашем примере мы будем использовать GCC и язык программирования C, потому что большинство читателей знакомы с этим конкретным языком и компилятором. Файл конфигурации gcc.cgpr выглядит так:

```
configuration project GCC is
   package Compiler is
      for Driver ("C") use "/usr/gnat/bin/gcc";
      for Leading_Required_Switches ("C") use ("-c");
      for Object_File_Suffix ("C") use ".o";
      for Dependency_Switches ("C") use ("-MMD","-MF","");
      for Include_Switches ("C") use ("-I");
   end Compiler;
   package Naming is
      for Spec_Suffix ("C") use ".h";
      for Body_Suffix ("C") use ".c";
   end Naming;
end GCC;
```

Этот простой конфигурационный файл достаточен для компиляции файлов C и достаточно понятен. Пакет «Compiler» определяет исполняемый файл компилятора с необходимыми ключами командной строки и суффикс объектного файла. Пакет «Naming» вводит обычную схему именования для файлов C.

Gprbuild имеет встроенный механизм зависимостей, который позволяет избежать ненужной перекомпиляции, когда соответствующие исходные файлы не изменились. Разумеется, желательно использовать этот механизм. Мы достигаем этого здесь, установив переменную «Dependency_Switches», которая даёт параметры командной строки, которые запускают генерацию файлов зависимостей, параллельно с компиляцией. Существует также переменная «Dependency_Driver», которая может быть установлена, если вы предпочитаете генерировать файл зависимостей независимо от вызова компилятора. Если ваш компилятор поддерживает вывод зависимостей с помощью синтаксиса Makefile, этот механизм является простым способом получения эффективной инкрементной компиляции для данного языка.

Другие части конфигурационного файла, такие как те, которые описывают процесс связывания, будут описаны в будущем Gem.

Теперь, учитывая проект C, для его описания достаточно следующего файла main.gpr:

```
project Main is
   for Languages use ("C");
end Main;
```

Как указано, gprbuild необходимо передать как файл проекта, так и файл конфигурации:

```
gprbuild --config=gcc.cgpr -P main.gpr
```

Мы уже описали, что если параметр «--config» опущен, gprbuild автоматически создает подходящий файл конфигурации. Он делает это с помощью инструмента gprconfig. При выполнении gprconfig считывает базу знаний, описывающую множество компиляторов с их параметрами, их применимость к различным платформам и т. Д. Учитывая эти знания, gprconfig определяет, какой компилятор доступен в системе. Затем Gprconfig может создать файл конфигурации, содержащий только описания доступных и релевантных компиляторов.

База знаний по умолчанию уже содержит полные описания компиляторов GNAT и gcc и некоторые другие. В следующем Gem мы увидим другие детали gprbuild, которые можно настроить. В третьем Gem этой серии мы увидим, как добавить компилятор в базу знаний.

Связанный со статьёй текст программы

Attached Files отсутствуют

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Йоханнес (Johannes) имеет степень инженера в Ecole Centrale Paris, степень магистра компьютерных наук (Masters in Computer Science) из Технического университета в Дрездене (Technical University of Dresden), Германия и кандидатскую диссертацию по программе (a PhD in Program Verification), полученную в Университете Парижа 11 (University Paris 11). Он присоединился к AdaCore в 2011 году и работает главным образом в технологии статического анализа CodePeer и исследовательский проект Hi-Lite, целью которого является объединение модульных тестов и формальной проверки.

Обсуждение…

Один ответ на «Gem # 104: Gprbuild и файлы конфигурации - часть 1»

1. 29 апреля 2011 года в 14:03

Ада сказала:

Я люблю учиться Ada с тех пор, как друг сказал мне, насколько она хороша. И я полагаюсь на любое учреждение, которое отправит меня на стипендию.