Gem #108: Функция gprbuild и конфигурационные файлы - Часть 2

Автор: Johannes Kanig, AdaCore

Краткое содержание: Данный Gem - второй в серии из трёх публикаций, нацеленных на описание gprbuild. В этой публикации будет пояснён процесс конфигурации данного инструмент с помощью конфигурационных файлов.

Давайте начнём…

В первом Gem этой серии мы вспомнили, что gprbuild – это инструмент для автоматического управления процессом сборки проекта Ada. Преимущество gprbuild перед другими инструментами заключается в том, что он не ограничивается Ada и на самом деле легко настраивается, а именно путём предоставления файла конфигурации. В предыдущем Gem мы дали пример файла конфигурации, который даёт необходимую информацию для определения компилятора C. Напомним вызов gprbuild через командную строку с пользовательским файлом конфигурации:

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

где main.gpr - это файл проекта, содержащий C-файлы для компиляции и gcc.cgpr - это файл конфигурации, который рассмотрен в предыдущем Gem.

В этом Gem мы показываем общую структуру файла конфигурации и, в частности, описываем несколько атрибутов, которые не были описаны в предыдущем Gem.

Файл конфигурации состоит из проекта конфигурации, который представлен следующим образом:

```
configuration project is
    --  The content of the configuration project goes here
    --  ...
end ;
```

Такой проект конфигурации содержит ряд глобальных атрибутов и несколько пакетов. Глобальные атрибуты:

```
  * the attribute Default_Language, which will be used by gprbuild if the
    project does not define any programming language to be used;
  * a number of attributes related to libraries, shared libraries and archives.
* атрибут Default_Language задает язык программировани, который  будет
использоваться gprbuild по умолчанию, если проект явно не определяет какой-либо
язык программирования для использования;
* ряд атрибутов, связанных с библиотеками, разделяемыми библиотеками и
архивами.
```

Мы уже видели использование пакета `Naming`, который содержит атрибуты, относящиеся к схемам именования соответствующих языков. Можно указать суффикс для файлов спецификаций и файлов реализации (подходящих для таких языков, как C и Ada), а также получить имя файла из названия модуля для языков, имеющих концепцию модуля (в настоящее время Ada). Это может быть достигнуто с использованием атрибутов `Casing` и `Dot_Replacement`.

```
package Naming is
   for Spec_Suffix ("Ada") use ".ads";
   for Body_Suffix ("Ada") use ".adb";
   for Casing ("Ada") use "lowercase";
   for Dot_Replacement use "-";
end Naming;
```

В более ранней версии Gem мы представили большинство возможных атрибутов для пакета `Compiler`:

```
package Compiler is
   for Driver ("Ada") use "/.../bin/gcc";
   for Required_Switches ("Ada") use ("-c", "-x", "ada", "-gnatA");
end Compiler;
```

Существует также ряд параметров, связанных с файлами конфигурации, передаваемыми компилятору, хотя мы не будем подробно останавливаться на них.

Некоторые компиляторы, такие как gcc, способны генерировать информацию о зависимостях, которая может быть использована gprbuild для обеспечения более эффективного процесса инкрементной перекомпиляции. Соответствующие атрибуты являются частью пакета `Compiler`, а для gcc правильные значения следующие:

```
for Dependency_Switches ("C") use ("-Wp,-MD,");
for Dependency_Driver ("C") use ("gcc", "-E", "-Wp,-M", "");
```

Атрибут Dependency_Switches даёт аргументы командной строки, чтобы заставить, компилятор сгенерировать зависимости и одновременно скомпилировать исходный файл, в то время как атрибут Dependency_Driver указывает командную строку, которая только генерирует файл зависимостей, без перекомпиляции. В обоих случаях компилятор должен сгенерировать файл с суффиксом «.d», содержащий строки формы:

```
b.o: b.c b.h a.h c.h
```

Имя файла перед двоеточием – это имя объектного файла, а список файлов после двоеточия – это список исходных файлов, от которых зависит этот объектный файл. Если какой-либо из этих исходных файлов является более свежим, чем объектный файл (gprbuild проверяет это путем сравнения метки времени), объектный файл создаётся заново.

Пакет компилятора также содержит список параметров для указания сведений о пути поиска источника. Самый простой вариант-атрибут Include_Switches, указывающий параметр, который необходимо передать компилятору:

```
for Include_Switches ("C") use ("-I");
```

Если существует много исходных каталогов, командная строка может быть слишком длинной. Лучшей альтернативой является использование переменной среды, которая содержит путь поиска, разделённый двоеточиями:

```
for Include_Path ("Ada") use "ADA_INCLUDE_PATH";
```

Однако для крупных проектов этого может быть недостаточно; атрибут Include_Path_File определяет переменную среды, которую можно использовать, чтобы указать имя текстового файла, содержащего список путей поиска для источников:

```
for Include_Path_File ("Ada") use "ADA_PRJ_INCLUDE_FILE";
```

Если присутствует несколько или все атрибуты, связанные с исходными каталогами, gprbuild выбирает в соответствии со следующими предпочтениями: `Include_Path_File` используется, если он определён, иначе `Include_Path`, если он определён, и `Include_Switches` используется в качестве последнего средства.

Атрибутом связывающий имена модулей объектов с именами файлов является атрибут `Mapping_File_Switches`, который определяет параметр компилятора для указания файла сопоставления. Файл сопоставления похож на файл пути включения, но он непосредственно

содержит сопоставление имён объектов с именами файлов, что более эффективно, особенно при наличии удалённых исходных каталогов.

```
for Mapping_File_Switches ("Ada") use ("-gnatem=");
```

Пакет `Binder` определяет, как вызвать редактор связей. Как обычно, можно указать исполняемый файл и обязательные параметры, но есть также два варианта, аналогичные тем, что для пути поиска источника, которые позволяют указывать путь поиска объекта.

```
package Binder is
   for Driver ("Ada") use "/.../gprbind";
   for Required_Switches ("Ada") use ("--prefix=");
   for Objects_Path ("Ada") use "ADA_OBJECTS_PATH";
   for Objects_Path_File ("Ada") use "ADA_PRJ_OBJECTS_FILE";
end Binder;
```

Наконец, в пакете `Linker` можно указать драйвер (исполняемый файл) компоновщика, все необходимые параметры и параметры для создания файла `map`:

```
package Linker is
   for Driver use "g++";
   for Map_File_Option use "-Wl,-Map,";
end Linker;
```

В предыдущем Gem мы также упомянули, что gprbuild может создать конфигурационный файл автоматически, когда он не предусмотрен, содержащий все необходимые определения для предпочитаемого компилятора для каждого языка данного проекта. Эта автоматическая генерация выполняется с помощью инструмента gprconfig, который получает необходимую информацию из набора XML-файлов, называемых базой знаний. В следующей и последнем Gem этой серии мы узнаем, как использовать gprconfig, а также как добавить компилятор в базу знаний, чтобы он мог быть выбран автоматически gprconfig.

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.

Last Updated: 11/24/2017
Posted on: 6/17/2011

Обсуждение…