Gem #65: Утилита управления компиляцией приложений в GNAT: gprbuild

Автор: Emmanuel Briot, AdaCore

Краткое содержание: Gem Ada #65 - `gprbuild` - это новая функция построения программ, заменившая `gnatmake`. Она работает со множеством языков, автоматически контролирует зависимость по исходному коду и уменьшает необходимость повторной компиляции. В данном Gem #65 в общих чертах описывается работа `gprbuild`.

Давайте начнём…

GNAT обеспечивает большую гибкость при компиляции приложений. Традиционный метод заключается в использовании `gnatmake` в командной строке, указывая список исходных каталогов с помощью параметров `"-I"` и позволяя `gnatmake` решать, какие файлы необходимо перекомпилировать. Одним из недостатков этого подхода является то, что он работает только для файлов Ada. Если ваше приложение использует источники, написанные на других языках (что, вероятно, относится к большинству серьёзных приложений), то вам нужна какая-то оболочка вокруг `gnatmake`. Обычно это делается с помощью файла Makefile: сначала вы перекомпилируете C, C++ и другие исходные файлы, используя стандартные методы Makefile, затем вызовите `gnatmake` для обработки источников Ada и, наконец, (хотя это также может быть сделано как часть `gnatmake`) bind и link ваше приложение.

`Gnatmake` - это то, что мы называем строитель (a builder). Его роль состоит в том, чтобы изучить все исходные файлы в вашем приложении и решить, какие из них должны быть перекомпилированы. Затем он полагается на другие инструменты для выполнения фактической компиляции (`gcc`), bind (`gnatbind`) и link (`gnatlink`).

Несколько релизов назад, новый инструмент под названием `gprbuild` был введён в технологии GNAT. Это также конструктор, повторно использующий и расширяющий `gnatmake` для поддержки многоязычных приложений. Это означает, что он знает, как обрабатывать C, C++ и многие другие языки, и способен определить, какие исходные файлы для каждого языка должны быть перекомпилированы. Одно из преимуществ заключается в том, что вам больше не нужно зависеть от файла `Makefile` для компиляции этих исходных файлов перед привязкой и связыванием.

`Gprbuild` был тщательно разработан таким образом, что не было сделано жёсткого кодирования для любого из языков. Вместо этого вся информация хранится в файле конфигурации (обычно с расширением `".cgpr"`): как скомпилировать файл, как решить, является ли он актуальным, какие файлы генерируются в результате компиляции и так далее.

Эти файлы конфигурации имеют синтаксис, аналогичный синтаксису самих файлов проекта. Их можно редактировать вручную по мере необходимости и поддерживать в системе управления версиями, как и для любого другого компонента среды.

Однако разработка файла конфигурации не всегда проста, особенно при использовании различных цепочек компиляторов (таких как GNAT для файлов Ada и Diab для C). В частности, фазу связи становится сложнее описать. Добавьте к этому поддержку нескольких платформ, кросс-компиляции, библиотек, и других тонкостей, и сложность быстро возрастает.

AdaCore уже позаботился о множестве возможных компиляторов и комбинаций, так что вам не придётся переделывать эту сложную настройку самостоятельно. Мы создали базу знаний, которая описывает конфигурацию для различных сценариев.

GNAT поставляется с дополнительным инструментом командной строки, называемым gprconfig, процесс которого обычно порождается прозрачно gprbuild, поэтому вам не нужно вызывать его напрямую в большинстве случаев. Роль gprconfig состоит в том, чтобы создать файл конфигурации на основе определённого набора компиляторов, получая информацию из базы знаний.

Теперь, когда мы описали общую организацию вещей, давайте покажем конкретный пример.

Предположим, ваше заявление состоит из двух файлов, bar.adb и foo.c, со следующим кодом (сам код не очень интересен, так как он ничего не будет делать):

```
procedure Bar is
   procedure Foo;
   pragma Import (C, Foo);
begin
   Foo;
end Bar;
```

```
int foo () {}
```

gprbuild работает только с файлами проекта (например, нет параметров командной строки для указания исходных каталогов). Поэтому вам нужно написать всю необходимую информацию в файле проекта. Основные элементы информации, которые необходимо указать в файле проекта: языки (C и Ada в нашем случае), исходные каталоги и исходные файлы (заданы неявным образом, в нашем случае) и главные исходные файлы (bar.adb в нашем случае). Таким образом, файл проекта default.gpr содержит следующие строки:

```
project Default is
   for Languages use ("C", "Ada");
   for Main use ("bar.adb");
end Default;
```

На данный момент все настроено, и мы можем просто создать gprbuild с помощью команды:

```
gprbuild –Pdefault
```

Так как мы не указали файл конфигурации через --config коммутатор, gprbuild автоматически сгенерирует его, порождая процесс gprconfig и сообщая ему, что мы хотим найти первые доступные компиляторы Ada и C на пути. Затем gprconfig создаст файл конфигурации с именем auto.cgpr, который gprbuild, в свою очередь, используются для построения наших исполняемых файлов.

При повторном запуске той же команды немедленно повторная компиляция не выполняется, даже для файла C.

Обратите внимание также, как gprconfig выбирает первый соответствующий компилятор на пути поиска PATH для каждого языка. Если вы хотите принудительно использовать определенные компиляторы (которые могут даже не быть на вашем PATH), вам нужно будет вручную запустить gprconfig и передать ему параметр --config. gprconfig создает файл конфигурации, который Вы затем можете передать при вызове gprbuild.

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

Сведения об авторе отсутствуют.

Last Updated: 10/13/2017
Posted on: 5/18/2009

Обсуждение…

Один ответ на Gem Ada #65: gprbuild”

1. 8-го июля 2009 в 23:09

Brian сказал:

gnatmake уходит в какой-то момент (т.е. “заменяет”), или будет сохранённым для тех из нас, кто все ещё считает его полезным? Не всегда стоит создать файл проекта для всего, что Вы хотите скомпилировать - как примеры в этих Gem. Это особенно полезно для маленького, но не тривиального кода, как основная процедура и один или два пакета (возможно вместе с gnatchop от предыдущего Gem Ada #64 :-).

(Надо надеяться, это может быть сохранено, как gnatkr - который я не использую, но помню. Вне темы, но я никогда не понимал, почему библиотеки - все еще Krunched.)