Gem #155: Настройка базы данных GPRBuild для работы с новым языком.

Автор: Vincent Celier - AdaCore

Краткое содержание: Базу данных GPRBuild можно дополнить файлом XML, в котором описаны характеристики нового языка. Это позволит файлу проекта, в котором объявлен данный язык автоматически вызывать компилятор для получения доступа к источникам данного языка и всем необходимым параметрам.

Давайте начнём…

В Gem #152, мы описали, как определить новый язык внутри файла проекта. Это полезно, когда вы используете язык, очень редко и когда у вас есть только один файл проекта на языке. Но когда вы часто используете язык, и у вас есть несколько проектов, использующих этот язык, он становится довольно громоздким.

К счастью, есть способ указать `gprbuild` все характеристики языка без необходимости повторять их в каждом проекте, который использует язык. Фактически, это метод, используемый для поддержки **Ada**, **C**, **C++** и **Fortran** изначально в `gprbuild`.

gprconfig XML database

Как мы обсуждали в предыдущем Gem, `gprbuild` читает определение языка и его инструменты поддержки из файла проекта. Фактически, он также использует второй (чаще всего неявный) файл, называемый файлом проекта конфигурации или просто файлом конфигурации.

Файл конфигурации использует тот же синтаксис, что и сами файлы проекта, и объединяется со всеми файлами проекта, которые анализируются `gprbuild`. Таким образом, цель состоит в том, чтобы изменить содержимое файла конфигурации, чтобы добавить информацию для нового языка там, а не в каждом проекте в дереве проектов.

Когда `gprbuild` вызывается без указанного файла проекта конфигурации (указывается переключателями `--config=` или `--autoconf=`), и нет файла проекта конфигурации по умолчанию (по умолчанию `default.cgpr`), `gprbuild` вызывает `gprconfig` для создания файла проекта конфигурации, а затем использует вновь созданный файл. Это называется автоматической конфигурацией.

Для создания этого файла проекта конфигурации `gprconfig` использует базу данных XML. По умолчанию используемые XML-файлы находятся в каталоге `<prefix>/share/gprconfig`, где вы найдете такие файлы, как `compilers.xml, c.xml` и `gnat.xml`.

Также можно указать, что `gprconfig` должен учитывать другие XML-файлы в другом каталоге. Это делается через `--db` ключ:

```
--db dir  (Parse dir as an additional knowledge base)
          (Dir – Это директорий синтаксического анализа как
           дополнительная база знаний)
```

Формат этих XML-файлов описан в Руководстве пользователя `GPRbuild`, в 2.2.6 база знаний `GPRconfig`.

Создание XML-файла

Так, для описания характеристик языка "New_Lang" мы создадим XML-файл new_lang.xml в каталоге "db".

XML-файл в gprconfig базе данных должен запуститься с:

```
<?xml version="1.0" ?>
<gprconfig>
```

и заканчивается:

```
</gprconfig>
```

Для каждого компилируемого языка должны быть тег <compiler_description> и тег <configuration>.

Именование компилятора

Первая часть работы, проделанной gprconfig, - определить, какие компиляторы установлены. Для этого он использует информацию из узлов <compiler_description> в любом из XML-файлов. Эти узлы должны объяснить, как найти исполняемый файл компилятора в любом месте пути, а затем извлечь из него информацию, такую как его версия и список поддерживаемых языков.

Тег <compiler_description> содержит несколько дочерних тегов. Некоторые из них являются обязательными: <name>, <executable>, <version> и <languages>.

В нашем XML файле newlang.xml , в теге compiler_description будет:

```
<compiler_description>
  <name>NEW_LANG</name>
  <executable>nlang</executable>
  <version>1.0</version>
  <languages>New_Lang</languages>
</compiler_description>
```

указывает, что он описывает компилятор NEW_LANG для языка "**New_Land**", что драйвер компилятора - "nlang" и что он имеет версию 1.0.

В общем, номер версии не жестко закодирован, как здесь, а является результатом запуска исполняемого файла со специальным переключателем, а затем разбора вывода с помощью регулярных выражений.

См. компиляторы файлов .xml в дистрибутиве gprbuild для некоторых примеров.

Описание характеристик языка

После этого первого прохода у gprconfig есть полный список всех доступных в системе компиляторов. Основываясь на потребностях проектов (в частности, какие языки программирования используются и цели), он попытается найти набор совместимых компиляторов и использовать их для создания файла конфигурации.

Полный контроль над атрибутами, которые необходимо добавить в файл конфигурации, осуществляется с помощью узла <configuration> в файлах XML.

Тегу <configuration> нужны два дочерних тега: <compilers> и <config>.

Тег <compilers> указывает различные языки/компиляторы/версии, к которым применяется данная конфигурация. Здесь нам нужно только указать, что имя «NEW_LANG».

```
<compilers>
  <compiler name="NEW_LANG">
</compiler>
```

Тег <config> указывает фрагменты, которые необходимо включить в файл проекта конфигурации. Здесь нам нужно только иметь именование пакетов и компилятор.

Если в разных узлах <configuration> для одного и того же пакета имеется несколько "блоков", gprconfig автоматически объединяет эти блоки в пакет в созданном файле конфигурации.

Итак, наш XML-файл new_lang.XML будет содержать:

```
<?xml version="1.0" ?>
  <gprconfig>
    <compiler_description>
      <name>NEW_LANG</name>
      <executable>nlang</executable>
      <version>1.0</version>
      <languages>New_Lang</languages>
    </compiler_description>
   <configuration>
      <compilers>
        <compiler name="NEW_LANG">
      </compiler>
      <config>
  package Naming is
     for Body_Suffix ("New_Lang") use ".nlng";
  end Naming;
package Compiler is
     for Driver ("New_Lang") use "nlang";
     for Leading_Required_Switches ("New_Lang") use ("-c");
     for Dependency_Kind ("New_Lang") use "Makefile";
     for Dependency_Switches ("New_Lang") use ("--dependencies=");
     for Include_Switches ("New_Lang") use ("-I", "");
  end Compiler;
      </config>
    </configuration>
  </gprconfig>
```

Использование автоконфигурации gprbuild для нового языка

Чтобы использование автоконфигурации в gprbuild для нового языка было успешным, нам нужно убедиться, что компилятор "nlang" находится на пути PATH.

Если у нас есть файл проекта prj.gpr в текущем рабочем каталоге, который содержит:

```
project Prj is
   for Languages use ("New_Lang");
end Prj;
```

и источник **New_Lang** foo.nlng, то вызов:

```
gprbuild prj.gpr --db db
```

вызовет компилятор для `foo.nlng`:

```
$ gprbuild prj.gpr --db db
nlang -c foo.nlng
$
```

Теперь у нас есть способ скомпилировать источники нашего языка **New_Lang** без необходимости повторять все характеристики языка в каждом из файлов проекта с источниками **New_Lang**. Однако нам все равно нужно вызвать `gprbuild` с помощью ключа `--db`. Могли бы мы сделать лучше?

Включение файлов XML в базу данных gprconfig по умолчанию

Да мы можем! Если мы просто скопируем наш XML-файл `new_lang.xml` в базу данных XML `gprconfig` по умолчанию `<prefix>/share/gprconfig`, язык **New_Lang** будет автоматически учитываться `gprconfig`, и нам больше не нужно будет вызывать `gprbuild` (или `gprconfig`) с ключом `--db`:

```
$ gprbuild prj.gpr
nlang -c foo.nlng
$
```

Резюме

Мы описали способ автоматической настройки нового языка в `gprbuild` с помощью XML-файла.

Конечно, этот пример очень прост. Полная документация XML-файлов `prconfig` находится в руководстве пользователя `GPRbuild` в разделе (2.2.6 база знаний `GPRconfig`) и его подразделах.

Итак, если вы заинтересованы в описании ваших новых языков через XML-файлы, мы рекомендуем Вам прочитать его и изучить различные XML-файлы в `<prefix>/share/gprconfig`.

Связанный со статьёй текст программы

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

Об авторе

Автор: Vincent Celier - AdaCore

Винсент Селье провел двадцать лет во французском флоте в качестве офицера по радарам и компьютерного оборудования. Он ушел в отставку в 1988 году в ранге командира и присоединился к CR2A, фирме по разработке программного обеспечения, где он был одним из авторов Технического отчета ISO ExtrA (Extensions Temps Reel en Ada). В 1994 году он эмигрировал в Ванкувер, Канада, чтобы работать над CAATS, Канадской автоматизированной системой воздушного движения,

большой системой, написанной на языке Ада. Он присоединился к AdaCore в 2000 году. Он является основным разработчиком менеджера проектов и `gprbuild`, с многоязычной поддержкой разработчика.

Last Updated: 10/13/2017
Posted on: 12/9/2013

Обсуждение…