Gem #59: Генерация связей с заголовочными файлами C в Ada.

Автор: Arnaud Charlet, AdaCore

Краткое содержание: Gem Ada #59 - Одна из наиболее сложных областей Ada, с которой у разработчиков часто возникают проблемы, - взаимодействие Ada с другими языками. Несмотря на то, что в Ada доступны удобные и полноценные возможности создания интерфейсов для работы с другими компилируемыми языками, такими как C, C++ и Fortran, написание связующего кода для использования масштабных и комплексных API может вызывать трудности и приводить к ошибкам.

В данном Gem будет продемонстрирован новый инструмент AdaCore, автоматизирующий генерацию интерфейсов для заголовочных файлов языка «C» с помощью компилятора.

Давайте начнем…

GNAT теперь идёт с новым экспериментальным обязательным генератором для языков «C» и заголовков C++. Генератор предназначен, чтобы сделать 95% утомительной работы генерации спецификаций Ada от заголовочных файлов языков C++ или «C». Обратите внимание на то, что это - все ещё не вся работа по генерации, нет разработанного средства, чтобы генерировать 100% результат, полученные спецификации Ada проверяют и исправляют при необходимости.

Сгенерированный код использует синтаксис Ada 2005, что упрощает взаимодействие с другими языками по сравнению с предыдущими версиями Ada, в частности, путём создания предложений `limited with` и анонимных типов доступа.

В этом Gem #59 мы будем фокусироваться на генерации привязок к языку «C». Привязки к языку C++ будут рассмотрены в будущем Gem #60 и #61.

Запуск генератора привязки

Генератор связывания является частью компилятора GCC, который поставляется с последними версиями GNAT и может быть вызван с помощью переключателя `-fdump-ada-spec`, который генерирует специальные файлы Ada для файлов заголовков, указанных в командной строке, и все необходимые файлы заголовков по этим файлам транзитивно. Например:

```
$ g++ -c -fdump-ada-spec -C /usr/include/time.h
$ gcc -c -gnat05 *.ads
```

генерирует, под GNU/Linux, следующими файлами: `time_h.ads, bits_time_h.ads, stddef_h.ads, bits_types_h.ads,` которые соответствуют файлам `/usr/include/time.h,/usr/include/bits/time.h`, и т.д...., и затем компилируют эти спецификации Ada в режиме Ada 2005 года.

Переключатель `-C` говорит GCC извлекать комментарии из заголовков и пытаться генерировать соответствующие комментарии Ada.

Если Вы хотите генерировать единственный файл Ada а не переходное закрытие, Вы можете использовать переключатель `-fdump-ada-spec-slim` вместо этого.

Обратите внимание на то, что мы рекомендуем, если это возможно, использовать G ++ драйвер, чтобы генерировать привязку, даже для большинства заголовков языка «C», так как это в целом генерирует лучшие спецификации Ada. Если G ++ не работает над Вашими заголовками языка «C» из-за несовместимостей между языками «C» и C++, то Вы можете возвратиться к использованию GCC вместо этого.

Для примера лучшей привязки, сгенерированной от фронтэнда-компилятора C++, имя параметров (когда доступно) на самом деле проигнорировано фронтэндом-компилятора «C». Рассмотрите следующий заголовок языка «C»:

```
extern void foo (int variable);
```

С фронтэндом-компилятора «C» проигнорирована "переменная", и вышеупомянутое обработано как:

```
extern void foo (int);
```

создание следующей спецификации подпрограммы:

```
procedure foo (param1 : int);
```

С фронтэндом C++ имя доступно, и мы генерируем:

```
procedure foo (variable : int);
```

В некоторых случаях сгенерированная привязка будет больше завершенным или более значимым при определении макросов, которые Вы можете сделать через переключатель `-D`. Это, например, имеет место с Xlib.h под GNU/Linux:

```
g++ -c -fdump-ada-spec -DXLIB_ILLEGAL_ACCESS -C /usr/include/X11/Xlib.h
```

Вышеупомянутое генерирует больше полной привязки, чем прямой вызов без переключателя-`DXLIB_ILLEGAL_ACCESS`.

В других случаях не возможно проанализировать заголовочный файл автономным способом, потому что файл включает дополнительно другие файлы, которые должны быть вставлены сначала. В этом случае решение состоит в том, чтобы создать маленький заголовочный файл, включая необходимый `#include` и возможные `#define` директивы. Например, чтобы генерировать привязку Ada для `readline/readline.h`, Вы сначала должны включать stdio.h, таким образом, Вы можете создать файл со следующими двумя строками, например, `readline1.h`:

```
#include <stdio.h>
#include <readline/readline.h>
```

и затем генерируйте привязку Ada от этого файла:

```
$ g++ -c -fdump-ada-spec readline1.h
```

В будущем Gem #60 и #61 мы будем говорить о генерации подобной привязки Ada для заголовков C++.

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

Данные об авторе отсутствуют.

Last Updated: 10/13/2017

Обсуждение…

8 ответов на "Gem # 59: Создание привязок Ada для заголовков C"

1. 24 февраля 2009 года в 12:37

qunying сказал:

Какая версия gnat / gcc начинает пользоваться этой функцией?

2. 24 февраля 2009 года в 9:25

Арно Шарлет сказал:

GNAT 6.2.1 предоставляет эту функцию

3. 24 февраля 2009 года в 19:24

Пол Пирсон сказал:

Это может быть очень удобно для связи с некоторыми языковыми библиотеками скриптов, такими как lua.

Как `-fdump-ada-spec` обрабатывает функциональные макросы? Будет ли g ++ делать это «лучше», чем gcc?

4. 24 февраля 2009 года в 19:48

Арно Шарлет сказал:

Я вижу, что сейчас мы переходим на продвинутые темы, вероятно, хорошие материалы для будущих Gem :-)

функциональные макросы - настоящая проблема, потому что им не хватает информации о параметрах (и потому, что они могут включать любой произвольный код).

В настоящее время генератор связывания будет генерировать псевдо-код Ada для функционально-подобных макросов, которые могут быть скопированы / вставлены и преобразованы в встроенные функции Ada, добавив вручную соответствующие типы, по крайней мере в простых случаях, что представляет собой, возможно, 80% типичного использования макросов arg.

Например, если вы считаете следующий макрос C:

```
#define func1 (a, b) (a + b)
```

он будет «переведен» на:

```
- arg-macro: function func1 (a, b)
```

- вернёт `a + b`;

5. 24 февраля 2009 года в 19:49

Арно Шарлет сказал:

Я забыл упомянуть: использование gcc или g ++ для случая макросов здесь не будет иметь никакого значения.

6. 24 февраля 2009 года в 23:01

Пол Пирсон сказал:

Спасибо, Армо. Я знал, что функциональные макросы будут сложными (одна из причин, по которым они могут быть подвержены ошибкам в C). Похоже, что генератор привязок берет безопасный подход. Как Ада нравится! ;-)

Да, использование генератора привязки для небольшой, полезной библиотеки может сделать интересный Gem!

7. 25 февраля 2009 года в 19:03

qunying сказал:

Была ли функция включена в официальный GCC? Или новая версия GNAT GPL будет иметь эту функцию?

8. 25 февраля 2009 года в 19:07

Арно Шарлет сказал:

Официальная разработка GCC в настоящее время проводится по завершению работы ветки GCC 4.4, поэтому на этом этапе новый взнос не может быть осуществлён. После того, как дерево GCC будет открыто для вкладов, мы внесём эти изменения для включения.

GNAT GPL 2009 также будет содержать эту новую функцию.