Вариативность модулей в BlackBox Component Builder

(ТЕЭ 09.01.2021 3:36:57 +0300)

1. Введение

Под вариативностью модулей здесь понимается поддержка системой нескольких вариантов A[1], A[2], … модуля A на уровне инструментов и возможность выбора одного из вариантов A[i] в момент загрузки. Поддержка на уровне инструментов потребовалась для кросс-разработки и возможности поселить разные сборки в одном каталоге. Необходимый минимум — поддержка разных вариантов модуля Kernel. Предложенное ниже решение — один из способов достичь этой цели. Поддержка выбора варианта (кодового файла) при загрузке — сопутствующий эксперимент, основанный на имеющихся возможностях системы Блэкбокс.

Система позволяет загрузить любой из вариантов A1, A2, … модуля A для его клиента модуля Б при условии, что интерфейсы A1, A2, … с точки зрения Б одинаковы. Подробно данный вопрос разобран в диссертации "Separate Compilation and Module Extension" (http://oberoncore.ru/library/crelier_separate_compilation_and_module_extension).

Ниже приведен последний из трех вариантов решений. Одним из основных критериев отбора была минимизация вносимых в систему изменений.

2. Схема решения

Вариативность поддерживается только для реализации (кодового файла), но не интерфейса (символьного файла). Исключение — Kernel, для которого необходима полная вариативность.

Вариантный модуль представлен интерфейсным модулем (аналог DEFINITION) и несколькими модулями реализации. При компиляции для интерфейсного генерируется символьный и кодовый файл, для реализации только кодовый. Предполагается, что кодовый файл интерейсного модуля дает реализацию-заглушку, которая будет задействована, если загрузчик модулей вариантность не поддерживает или если отсутствует модуль реализации.

Компилятор отличает варианты от обычных модулей по расположению (полному имени)

файла с исходником. Если имя файла соответствует имени модуля в исходном тексте, то это либо интерфейсный модуль, либо обычный. Разницы между ними фактически нет. Если имя файла с исходником не соответствует имени модуля в исходном тексте, то это вариант реализации. Имя модуля в исходнике определяет интерфейсный модуль, имя файла исходника определяет положение кодового файла.

Test/Mod/Time64 — интерфейсный модуль

```
MODULE TestTime64;

   PROCEDURE GetTime* (OUT t: LONGINT);
   BEGIN
     t := 0
   END GetTime;

   PROCEDURE GetBias* (OUT b: LONGINT);
   BEGIN
     b := 0
   END GetBias;

END TestTime64.
```

Test/Mod/Time64Win — вариант-1 его реализации

```
MODULE TestTime64;
   …
   PROCEDURE GetTime* (OUT t: LONGINT) …
   PROCEDURE GetBias* (OUT b: LONGINT) …
END TestTime64.
```

Test/Mod/Time64Lin — вариант-2 его реализации

```
MODULE TestTime64;
   …
   PROCEDURE GetTime* (OUT t: LONGINT) …
   PROCEDURE GetBias* (OUT b: LONGINT) …
END TestTime64.
```

```
DevCompiler.CompileThis TestTime64 TestTime64Win TestTime64Lin
compiling "TestTime64"
  new symbol file   64   0
compiling "TestTime64" (from "TestTime64Win")
(writing code to TestTime64Win)   36   0
compiling "TestTime64" (from "TestTime64Lin")
(writing code to TestTime64Lin)   36   0
```

Для Kernel всегда генерируются кодовый и символьный, размещаемые по имени файла исходника.

```
DevCompiler.CompileThis HostKernel Host_winKernel
compiling "Kernel" (from "HostKernel")
(writing sym to HostKernel)
(writing code to HostKernel)   18004   4608
compiling "Kernel" (from "Host_winKernel")
(writing sym to Host_winKernel)
(writing code to Host_winKernel)   19668   4308
```

Поскольку интерфейсы у разных вариантов Kernel различаются, в момент компиляции нужно понимать, какой использовать. Вариант Kernel задается командой DevCPM.SetKernel:

```
PROCEDURE SetKernel (IN mod: ARRAY OF CHAR)
```

```
"DevCPM.SetKernel('HostKernel')"
```

```
!DevCompiler.CompileThis HostFiles◢
DevCPM: HostKernel -> Kernel
compiling "HostFiles"
(importing Kernel from HostKernel)   17600   588

!"DevCPM.SetKernel('Host_winKernel')"
!DevCompiler.CompileThis Host_winFiles◢
DevCPM: Host_winKernel -> Kernel
compiling "Host_winFiles"
(importing Kernel from Host_winKernel)   16924   1108
```

Линкер использует имя модуля, записанное в кодовом файле. Кодовый файл он отыскивает по имени в командной строке.

```
!Dev2Linker1.LinkElfExe Linux dev0m := HostKernel$+ Files …◢
Linking Kernel from HostKernel
…
!DevLinker.Link dos dev0m.exe := Host_winKernel$+ Files …◢
Linking Kernel from Host_winKernel
…
```

Для обозревателя репозитория DevRBrowser и карты программиста ipuiK293 изменений не требуется.

Чтобы определять вариант при загрузке требуется переопределить StdLoader.ThisObjFile.

```
PROCEDURE ThisObjFile (VAR name: ARRAY OF CHAR): Files.File;
```

Ниже приведен загрузчик CorebbStdLoaderV, который получает варианты из таблицы. Таблица читается из файла при пуске системы.

3. Изменения: рантайм, компилятор, обзор, компоновщик

Примечание: кроме внедрения вариативности в изменениях отмечено отключение правила "системные модули в корне, затем в System" — они читаются только из System.

3.1 Рантайм. Механизм загрузки

StdLoader🡺🡸 — правило "системные модули в System"
CorebbStdLoaderV🡺🡸 — копия StdLoader, только отличия
CorebbStdLoaderVMapper — выдача кодовых файлов, таблица выбора вариантов
CorehStdLoaderVSetupLin — активация загрузки таблицы из файла, заданного параметром BB_VTABLE
CorehStdLoaderVSetupWin — тоже для Win

3.2 Инструменты. Компилятор

DevCPM🡺🡸

DevCPT🡺🡸

DevAnalyzer🡺🡸

DevCompiler🡺🡸

3.3 Инструменты. Обзор

DevBrowser🡺🡸
StdDialog🡺🡸

3.4 Инструменты. Компоновщик

Dev2LnkBase🡺🡸
Dev2LnkLoad🡺🡸

DevLinker🡺🡸

CorebbStdLoaderTableConv — конвертер файлов таблиц

4. Примеры использования

Требуется:
- сборка ББ https://oberoncore.ru/projects/oct — внесены изменения из пункта 3;
- подсистема ipui — https://oberoncore.ru/bbcc/subs/ipui/;
- архив core-vmod.txt с модулями Corebb/Coreh — https://oberoncore.ru/library/temir_core-vmod.

В примере ниже используются варианты реализации модулей Kernel и TestLog для Windows/Linux. TestInit — вариант реализации Init содержит тестовый код для пускача системы с динамическим загрузчиком.

Данный подход применим к HostConInit/HostGuiInit/Host_winConInit/Host_winGuiInit, что позволяет исключить модули HostConStartup/HostGuiStartup/Host_winConStartup/Host_winGuiStartup из пускача системы, используя схему Init = пускач при соседстве разных сборок в одном каталоге.

Пример с TestLog показывает, как можно заменять хуки, когда их установка требуется при пуске системы (например, Dates.Hook). Дает ли такая замена какой-либо выигрыш — вопрос открытый.

Варианты могут быть использованы для подключения альтернативных реализаций стандартных вьюшек. Это еще один из способов подмены кодового файла стандартной реализации.

4.1 TestLog

Интерфейс TestLog
"ipuiK298.WriteThis('Test/Mod', 'Log.odc', '')"

```
MODULE TestLog;

   PROCEDURE String* (IN x: ARRAY OF CHAR); BEGIN END String;
   PROCEDURE Int* (x: LONGINT); BEGIN END Int;
   PROCEDURE Ln*; BEGIN END Ln;
   PROCEDURE Tab*; BEGIN END Tab;

END TestLog.
```

И варианты его реализации для Lin/Win:
"ipuiK297.MapFold2ParE; ipuiK298.WriteThis('Test/Mod', 'LogLin.odc', '')"
"ipuiK297.MapFold2ParE; ipuiK298.WriteThis('Test/Mod', 'LogWin.odc', '')"

Модуль-клиент TestLog
"ipuiK298.WriteThis('Test/Mod', 'ConLog.odc', '')"

```
MODULE TestConLog;

   IMPORT Log := TestLog;

   PROCEDURE Do*;
   BEGIN
      Log.String("Привет!"); Log.Ln
   END Do;

BEGIN
   Do
END TestConLog.
```

Статическая линковка

```
DevCompiler.CompileThis TestLog TestLogLin TestLogWin TestConLog
compiling "TestLog"
  new symbol file   12   0
compiling "TestLog" (from "TestLogLin")
(writing code to TestLogLin)   996   0
compiling "TestLog" (from "TestLogWin")
(writing code to TestLogWin)   780   8
compiling "TestConLog"
  new symbol file   48   0
```

Необходимые варианты реализации TestLog задаются при линковке

Linux

```
Dev2Linker1.LinkElfExe Linux test_log := TestLogLin TestConLog
Linking TestLog from TestLogLin
test_log written
"ipuiK489.Exec('lin')" chmod u+x ./test_log && ./test_log ; echo ВЫП ; read
```

Windows

```
DevLinker.LinkExe test_log.exe := TestLogWin TestConLog
Linking TestLog from TestLogWin
test_log.exe written   4608   860
"ipuiK489.Exec('win')" test_log.exe && pause
"ipuiK489.Exec('lin')" wine test_log.exe ; echo ВЫП ; read
```

Динамическая

Конвертер для редактирования таблиц:

```
DevCompiler.CompileThis CorebbStdLoaderVTableConv
CorebbStdLoaderVTableConv.Register
```

Вместо TestConLog необходим вариант TestInit пускового модуля системы — Init

```
"DevCompiler.CompileThisTextV('TestInit')"
MODULE Init;

   IMPORT Log := TestLog;

   PROCEDURE Do*;
   BEGIN
      Log.String("Привет!"); Log.Ln
   END Do;

BEGIN
   Do
END Init.
```

```
DevCompiler.CompileThis HostKernel Host_winKernel
```

Linux

```
"DevCPM.SetKernel('HostKernel')"
DevCompiler.CompileThis CorebbStdLoaderVMapper CorebbStdLoaderV
CorehStdLoaderVSetupLin
```

```
Dev2Linker1.LinkElfExe Linux test_log_dyn := HostKernel+ Files Utf8 HostConLinkedLog
```

HostConf HostFiles CorebbStdLoaderVMapper CorehStdLoaderVSetupLin CorebbStdLoaderV
"ipuiK489.Exec('lin')" chmod -v u+x ./test_log_dyn ; read

Таблица вариантов:
"ipuiK298.RewriteThis('', 'TestLogLin.vtab', 'vtab')"Init TestInit
TestLog TestLogLin

Запуск
"ipuiK489.Exec('lin')" ./test_log_dyn --BB_VTABLE=TestLogLin.vtab ; read

Windows
"DevCPM.SetKernel('Host_winKernel')"
DevCompiler.CompileThis CorebbStdLoaderVMapper CorebbStdLoaderV CorehStdLoaderVSetupWin

DevLinker.LinkExe test_log_dyn.exe := Host_winKernel+ Files Utf8 Host_winConLinkedLog Host_winConf Host_winFiles CorebbStdLoaderVMapper CorehStdLoaderVSetupWin CorebbStdLoaderV

Таблица вариантов:
"ipuiK298.RewriteThis('', 'TestLogWin.vtab', 'vtab')"Init TestInit
TestLog TestLogWin

Запуск
"ipuiK489.Exec('win')" ./test_log_dyn.exe --BB_VTABLE=TestLogWin.vtab && pause
"ipuiK489.Exec('lin')" wine ./test_log_dyn.exe --BB_VTABLE=TestLogWin.vtab ; read