Oml МультиОберон для LLVM Быстрый Старт

Oml это реализация компилятора МультиОберон с бэкендом LLVM. Использует подготовленную библиотеку на основе LLVM 5.0.
Версия 0.95 27-Jun-2020

Инсталляция.

Для Windows (цвет - для Windows):
For BlackBox 1.6
```
win_toinstall.vbs 16 <path-to-blackbox>
```
For BlackBox 1.7
```
win_toinstall.vbs 17 <path-to-blackbox>
```

Для Linux (цвет – для Linux):
glibc 2.15 или STT_GNU_IFUNC поддержка требуется, tinfo пакет нужен
Загрузите и установите https://blackbox.obertone.ru/download
```
tclsh lin_toinstall.tcl 17 <path-to-blackbox>
```

Как начать работу из Black Box

1.1. Предусловия.
Oml использует LLVM 5.0 в виде библиотеки LLVMT.dll.
2. Oml/Docu/Quick-Start.odc

3 Скомпилируйте доступ к LLVM:
```
^Q DevCompiler.CompileThis LlvmC LlvmForAArch64 LlvmForAMDGPU LlvmForARM
LlvmForBPF LlvmForHexagon LlvmForLanai LlvmForMips LlvmForMSP430 LlvmForNVPTX
LlvmForPowerPC LlvmForSparc LlvmForSystemZ LlvmForX86 LlvmForXCore LlvmNative
LlvmRefs
```

3 Скомпилируйте модули ниже:
```
^Q DevCompiler.CompileThis HostApi HostConLog HostTimes OmcCfgfile OmcTarget
OmcCRuntime OmcHooks OmcDialog OmcOPM OmcOPT OmcOPU OmcOPB OmcOPS OmcOPP OmcDump
OmcTester OmcParams OmcCommandParams OmcOdcSource OmcOdcTextReader OmcExtSource
OmcRuntimeStd OmcDialogStd OmcDialogConsole OmcCompiler OmcConsole OmcLoader
OmlOPG OmlOPL OmlOPF OmlOPC OmlOPV OmlParams OmlBackEnd OmlCompiler OmlLinker
```

2 Компиляция примеров

2.1. Скомпилируйте примеры для 32-bit:
```
^Q OmlCompiler.CompileThis OmtestHelloWorld OmtestFormats OmtestDateTime
OmtestMkTraps OmtestHeap
```
Ожидаемый результат в каталоге ~/Omtest/Clwe/: OmtestHelloWorld.ll OmtestHelloWorld.bc OmtestFormats.ll OmtestFormats.bc OmtestDateTime.ll OmtestDateTime.bc OmtestMkTraps.ll OmtestMkTraps.bc OmtestHeap.ll OmtestHeap.bc

2.2. Скомпилируйте примеры для 64-bit:
```
^Q OmlCompiler.CompileThis -64 OmtestHelloWorld OmtestFormats OmtestDateTime
OmtestMkTraps OmtestHeap
```
Ожидаемый результат в каталоге ~/Omtest/Clwr/: OmtestHelloWorld.ll OmtestHelloWorld.bc OmtestFormats.ll OmtestFormats.bc OmtestDateTime.ll OmtestDateTime.bc OmtestMkTraps.ll OmtestMkTraps.bc OmtestHeap.ll OmtestHeap.bc

3 Авто-компиляция Oml Shell

3.1 Скомпилируйте 32-битный доступ к LLVM:

```
^Q OmlCompiler.CompileThis LlvmC LlvmForAArch64 LlvmForAMDGPU LlvmForARM
LlvmForBPF LlvmForHexagon LlvmForLanai LlvmForMips LlvmForMSP430 LlvmForNVPTX
LlvmForPowerPC LlvmForSparc LlvmForSystemZ LlvmForX86 LlvmForXCore LlvmNative
LlvmRefs
```

3.2 Авто-компиляция 32-битной консоли OmlSh.

```
^Q OmlCompiler.CompileThis LlvmNative OmcCfgfile OmcTarget OmcCRuntime OmcDialog
OmcHooks OmcOdcTextReader OmcExtSource OmcDialogConsole OmcRuntimeStd OmcOPM
OmcOPT OmcOPB OmcOPU OmcOPS OmcOPP OmcTester OmcParams OmcConsole OmcDump
OmcShell OmlOPG OmlOPL OmlOPF OmlOPC OmlOPV OmlParams OmlBackEnd OmcLoader
OmcLoaderRoutines OmcObjLoader_Coff OmlBcLoader_Win :OmlSh

^Q OmlCompiler.CompileThis LlvmNative OmcCfgfile OmcTarget OmcCRuntime OmcDialog
OmcHooks OmcOdcTextReader OmcExtSource OmcDialogConsole OmcRuntimeStd OmcOPM
OmcOPT OmcOPB OmcOPU OmcOPS OmcOPP OmcTester OmcParams OmcConsole OmcDump
OmcShell OmlOPG OmlOPL OmlOPF OmlOPC OmlOPV OmlParams OmlBackEnd OmcLoader
OmcLoaderRoutines OmcObjLoader_Elf OmlBcLoader :OmlSh
```

3.3 Скомпилируйте 64-битный доступ к LLVM.

```
^Q OmlCompiler.CompileThis -64 LlvmC LlvmForAArch64 LlvmForAMDGPU LlvmForARM
LlvmForBPF LlvmForHexagon LlvmForLanai LlvmForMips LlvmForMSP430 LlvmForNVPTX
LlvmForPowerPC LlvmForSparc LlvmForSystemZ LlvmForX86 LlvmForXCore LlvmNative
LlvmRefs
```

3.4 Авто-компиляция 64-битной консоли OmlSh.

```
^Q OmlCompiler.CompileThis -64 LlvmNative OmcCfgfile OmcTarget OmcCRuntime
OmcDialog OmcHooks OmcOdcTextReader OmcExtSource OmcDialogConsole OmcRuntimeStd
OmcOPM OmcOPT OmcOPB OmcOPU OmcOPS OmcOPP OmcTester OmcParams OmcConsole OmcDump
OmcShell OmlOPG OmlOPL OmlOPF OmlOPC OmlOPV OmlParams OmlBackEnd OmcLoader
OmcLoaderRoutines OmcObjLoader_Coff OmlBcLoader_Win :OmlSh

^Q OmlCompiler.CompileThis -64 LlvmNative OmcCfgfile OmcTarget OmcCRuntime
OmcDialog OmcHooks OmcOdcTextReader OmcExtSource OmcDialogConsole OmcRuntimeStd
OmcOPM OmcOPT OmcOPB OmcOPU OmcOPS OmcOPP OmcTester OmcParams OmcConsole OmcDump
OmcShell OmlOPG OmlOPL OmlOPF OmlOPC OmlOPV OmlParams OmlBackEnd OmcLoader
OmcLoaderRoutines OmcObjLoader_Elf OmlBcLoader :OmlSh
```

4 Выгрузка компилятора Oml

```
^Q DevDebug.UnloadThis OmlCompiler OmlLinker OmlBackEnd OmlParams OmlOPV OmlOPC
OmlOPF OmlOPL OmlOPG OmcCompiler OmcDialogStd OmcRuntimeStd OmcOdcSource
OmcCommandParams OmcParams OmcTester OmcDump OmcOPP OmcOPS OmcOPU OmcOPB OmcOPT
OmcOPM OmcDialog OmcHooks OmcCRuntime OmcTarget OmcCfgfile Runner OLog HostTimes
Times OStrings
```

Как начать работу из Командной Строки.

1 Инсталляция

1. Предусловия.

Oml использует LLVM 5.0 в виде библиотеки LLVMT.dll.. Выполняйте все вышеперечисленные команды из корневого каталога Mob-master.

2 Компиляция примеров

```
Blwe\omlsh co OmtestHelloWorld
Blue/omlsh co OmtestHelloWorld
```

Новый символьный файл создается с соответствующим сообщением, затем OmtestHelloWorld.mod компилируется в 32-битовый Omtest/Clwe/HelloWorld.bc. Может быть задан не один, а список файлов для компиляции.

```
Blwe\omlsh ru OmtestHelloWorld
Blue/omlsh ru OmtestHelloWorld
```

Ru[n] – Выполнение. Выполнить 32-битовый OmtestHelloWorld.bc как динамически загруженный модуль с помощью оболочки Oml Shell.

```
Blwe\omlsh ex OmtestHelloWorld
Blue/omlsh ex OmtestHelloWorld
```

Ex[ecute]. Execute означает 32-битовую компиляцию и выполнение OmtestHelloWorld.mod с помощью оболочки Oml Shell.

```
Blwe\omlsh co OmtestHelloWorld: OmtestFormats: OmtestDateTime: OmtestMkTraps:
OmtestHeap:
Blue/omlsh co OmtestHelloWorld: OmtestFormats: OmtestDateTime: OmtestMkTraps:
OmtestHeap:
```

32-битовая компиляция всех перечисленных примеров выполняется командой выше.

```
Blwr\omlsh co OmtestHelloWorld
Blur/omlsh co OmtestHelloWorld
```

Новый символьный файл создается с соответствующим сообщением, затем OmtestHelloWorld.mod компилируется в 64-битовый Omtest/Clwr/HelloWorld.bc. Может быть задан не один, а список файлов для компиляции.

```
Blwr\omlsh ru OmtestHelloWorld
Blur/omlsh ru OmtestHelloWorld
```

Run Ru[n] – Выполнение. Выполнить 64-битовый OmtestHelloWorld.bc как динамически загруженный модуль с помощью оболочки Oml Shell.

```
Blwr\omlsh ex OmtestHelloWorld
Blur/omlsh ex OmtestHelloWorld
```

Ex[ecute]. Execute означает 64-битовую компиляцию и выполнение OmtestHelloWorld.mod с помощью оболочки Oml Shell.

```
Blwr\omlsh co +HostConLog OmtestHelloWorld: OmtestFormats: OmtestDateTime:
OmtestMkTraps: OmtestHeap:
Blur/omlsh co +HostConLog OmtestHelloWorld: OmtestFormats: OmtestDateTime:
OmtestMkTraps: OmtestHeap:
```

64-битовая компиляция всех перечисленных примеров выполняется командой выше.

3 Выполнение примеров

3.1. Самый простой пример - Hello, World

```
Blwe\omlsh ru OmtestHelloWorld
Blue/omlsh ru OmtestHelloWorld
```

Печать строковых, целых и действительных чисел

```
Blwe\omlsh ru OmtestFormats
Blue/omlsh ru OmtestFormats
```

3.2 Печать даты, времени и реализация задержек

```
Blwe\omlsh ru OmtestDateTime
Blue/omlsh ru OmtestDateTime
```

3.3 Трапы – обработка нештатных ситуаций

Простой Assert

```
Blwe\omlsh ru OmtestMkTraps -trap a
Blue/omlsh ru OmtestMkTraps -trap a
```

Простой Halt

```
Blwe\omlsh ru OmtestMkTraps -trap h
Blue/omlsh ru OmtestMkTraps -trap h
```

Деление на ноль

```
Blwe\omlsh ru OmtestMkTraps -trap z
Blue/omlsh ru OmtestMkTraps -trap z
```

Обращение к памяти по нулевому указателю

```
Blwe\omlsh ru OmtestMkTraps -trap p
Blue/omlsh ru OmtestMkTraps -trap p
```

3.4 Работа с динамической памятью и сборка мусора

```
Blwe\omlsh ru OmtestHeap
Blue/omlsh ru OmtestHeap
```

4 Набор примеров

Предусловия.

Для компиляции и линковки в бинарники требуются средства разработки для языка C. В данном пакете эти средства не поставляются и представлены не будут: ни Visual Studio, ни MinGW, ни CMake. Вы можете устанавливать средства разработки C или модифицировать скрипты сборки под установленные уже средства на Вашем компьютере. В скриптах используется следующее:

- gcc, ar – для lwe (oFront-Windows-32bit),
- clang – для lwr (oFront-Windows-64bit).

```
lwe_tomake
lue_tomake.sh
```

Создает все исполняемые файлы набора примеров для 32-бит

```
lwe_toclean
lue_toclean.sh
```

Удаляет все исполняемые файлы набора примеров для 32-бит

```
lwr_tomake
lur_tomake.sh
```

Создает все исполняемые файлы набора примеров для 64-бит

```
lwr_toclean
lur_toclean.sh
```

Удаляет все исполняемые файлы набора примеров для 64-бит

5 Выполнение примеров как приложений

5.1. Самый простой пример - Hello, World (64-bit)

```
Omtest\Clwr\OmtestHelloWorld
Omtest/Clur/OmtestHelloWorld
```

Печать строковых, целых и действительных чисел

```
Omtest\Clwr\OmtestFormats
```

```
Omtest/Clur/OmtestFormats
```

5.2 Печать даты, времени и реализация задержек

```
Omtest\Clwr\OmtestDateTime
Omtest/Clur/OmtestDateTime
```

5.3 Трапы – обработка нештатных ситуаций

Простой Assert

```
Omtest\Clwr\OmtestMkTraps -trap a
Omtest/Clur/OmtestMkTraps -trap a
```

Простой Halt

```
Omtest\Clwr\OmtestMkTraps -trap h
Omtest/Clur/OmtestMkTraps -trap h
```

Деление на ноль

```
Omtest\Clwr\OmtestMkTraps -trap z
Omtest/Clur/OmtestMkTraps -trap z
```

Обращение к памяти по нулевому указателю

```
Omtest\Clwr\OmtestMkTraps -trap p
Omtest/Clur/OmtestMkTraps -trap p
```

5.4 Работа с динамической памятью и сборка мусора

```
Omtest\Clwr\OmtestHeap
Omtest/Clur/OmtestHeap
```

6 Сборка бинарников OmlSh

Предусловия.

Для компиляции и линковки в бинарники требуются средства разработки для языка C. В данном пакете эти средства не поставляются и представлены не будут: ни Visual Studio, ни MinGW, ни CMake. Вы можете устанавливать средства разработки C или модифицировать скрипты сборки под установленные уже средства на Вашем компьютере. В скриптах используется следующее:

- gcc, ar – для lwe (oFront-Windows-32bit),
- clang – для lwr (oFront-Windows-64bit).

```
lwe_sh_tomake
lue_sh_tomake.sh
```

Создает все исполняемые файлы 32-бит оболочки OmlSh

```
lwe_sh_toclean
lue_sh_toclean.sh
```

Удаляет все исполняемые файлы 32-бит оболочки OmlSh

```
lwr_sh_tomake
lur_sh_tomake.sh
```

Создает все исполняемые файлы 64-бит оболочки OmlSh

```
lwr_sh_toclean
lur_sh_toclean.sh
```

Удаляет все исполняемые файлы 64-бит оболочки OmlSh

7 Набор тестов

```
lwe_tests_tomake
lue_tests_tomake.sh
```

И 64-bit

```
lwr_tests_tomake
lur_tests_tomake.sh
```

Подготовка всех тестовых файлов: OmtestOmcSimpleTest OmtestOmcStringsTest OmtestOmcSystemTest OmtestOmcImportsTest OmtestOmcExtensionsTest OmtestOmcBoundTest OmtestOmcAdvancedTest

```
lwe_tests_run
lue_tests_run.sh
```

И 64-bit
```
lwr_tests_run
lur_tests_run.sh
```
Запуск тестов компилятора
```
lwe_tests_jit_run
lue_tests_jit_run.sh
```
And 64-bit
```
lwr_tests_jit_run
lur_tests_jit_run.sh
```
Запуск тестов через LLVM Just-In-Time компилятор

Журнал изменений

may 2019 original MultiOberon pre-version 0.8 released
nov 2019 MultiOberon pre-version 0.9 released
jun 2020 MultiOberon pre-version 0.95 released

Use it and enjoy! - ЎЪsalos y disfrъtalos! - Bonne utilisation - Приятного использования - Powodzenia - Viel SpaЯ

Дагаев Дмитрий Викторович
dvdagaev@yahoo.com