Oml MultiOberon/LLVM Quick-Start

Oml is the instance of MultiOberon compiler with LLVM backend. Used prepared library with LLVM 5.0.
Version 24-May-2019

How to Start from Command Line.

1 Installation

1. Preconditions.
Oml doesn't use any other services. Process all the commands below from the Mob-master root dir.
Extract here from the Blwe.zip the Blwe catalog. Extract here from the Blwr.zip the Blwr catalog.

2 Compiling examples

```
Blwe\OmlSh co OmtestHelloWorld
```
A new symbol file is created first, then OmtestHelloWorld.mod is compiled to 32-bit Omtest/Clwe/HelloWorld.bc. A list of files can be compiled.

```
Blwe\OmlSh ru OmtestHelloWorld
```
Run 32-bit OmtestHelloWorld.bc with Oml Shell as dynamically loaded module.

```
Blwe\OmlSh ex OmtestHelloWorld
```
Execute means both 32-bit compile and run OmtestHelloWorld.mod with Oml Shell.

```
Blwe\OmlSh co OmtestHelloWorld: OmtestFormats: OmtestDateTime: OmtestMkTraps:
OmtestHeap:
```
The command above compiles all the examples listed for 32-bit.

```
Blwr\OmlSh co OmtestHelloWorld
```
A new symbol file is created first, then OmtestHelloWorld.mod is compiled to 64-bit Omtest/Clwr/HelloWorld.bc. A list of files can be compiled.

```
Blwr\OmlSh ru OmtestHelloWorld
```
Run 64-bit OmtestHelloWorld.bc with Oml Shell as dynamically loaded module.

```
Blwr\OmlSh ex OmtestHelloWorld
```
Execute means both 64-bit compile and run OmtestHelloWorld.mod with Oml Shell.

```
Blwr\OmlSh co OmtestHelloWorld: OmtestFormats: OmtestDateTime: OmtestMkTraps:
OmtestHeap:
```
The command above compiles all the examples listed for 64-bit.

3 Running the examples

3.1.The simplest Hello, World example
```
Blwe\OmlSh ru OmtestHelloWorld
```

Logging with char, int and real formats
```
Blwe\OmlSh ru OmtestFormats
```

3.2 Shows date, time and delay
```
Blwe\OmlSh ru OmtestDateTime
```

3.3 Traps handling abilities of runtime
Simple Assert
```
Blwe\OmlSh ru OmtestMkTraps -trap a
```
Simple Halt
```
Blwe\OmlSh ru OmtestMkTraps -trap h
```
Zero divide
```
Blwe\OmlSh ru OmtestMkTraps -trap z
```
Nil pointer dereference
```
Blwe\OmlSh ru OmtestMkTraps -trap p
```

3.4 Dynamic memory and garbage collector
```
Blwe\OmlSh ru OmtestHeap
```

4 Example set executives

In order to compile and link to binary executives the C-development environment is needed. I provide no Visual Studio or MinGW or CMake tools. Please, use external tools or modify scripts. I use the following:

- gcc, ar – for lwe,
- clang – for lwr.

```
lwe_tomake
```
Makes all the 32-bit executives of example set
```
lwe_toclean
```
Cleans all the 32-bit executives of example set
```
lwr_tomake
```
Makes all the 64-bit executives of example set
```
lwr_toclean
```
Cleans all the 64-bit executives of example set

5 Running executives

5.1.The simplest Hello, World example (64-bit)
```
Omtest\Clwr\OmtestHelloWorld.exe
```

Logging with char, int and real formats
```
Omtest\Clwr\OmtestFormats.exe
```

5.2 Shows date, time and delay
```
Omtest\Clwr\OmtestDateTime
```

5.3 Traps handling abilities of runtime
Simple Assert
```
Omtest\Clwr\OmtestMkTraps -trap a
```
Simple Halt
```
Omtest\Clwr\OmtestMkTraps -trap h
```
Zero divide
```
Omtest\Clwr\OmtestMkTraps -trap z
```
Nil pointer dereference
```
Omtest\Clwr\OmtestMkTraps -trap p
```

5.4 Dynamic memory and garbage collector
```
Omtest\Clwr\OmtestHeap
```

6 Making Compiled Shell Binaries

In order to compile and link to binary executives the C-development environment is needed. I provide no Visual Studio or MinGW or CMake tools. Please, use external tools or modify scripts. I use the following:

- gcc, ar – for lwe,
- clang – for lwr.

`lwe_compiler_tomake`
Makes all the 32-bit executive of OmlSh
`lwe_compiler_toclean`
Cleans all the 32-bit executive of OmlSh
`lwr_compiler_tomake`
Makes all the 64-bit executive of OmlSh
`lwr_compiler_toclean`
Cleans all the 64-bit executive of OmlSh

How to Start from Black Box

1 Installation

1.1. Preconditions.

Install LLVMT.dll from ~/Blwe to BlackBox.exe location
Oml uses LLVM 5.0 Services and LLVMT.dll.

2 Compile LLVM Services:

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

3 Compile the following modules:

```
^Q DevCompiler.CompileThis OmcTarget OmcCRuntime OmcHooks OmcDialog OmcOPM OmcOPT
OmcOPU OmcOPB OmcOPS OmcOPP OmcDump OmcParams OmcOdcSource OmcTxtSource
OmcRuntimeStd OmcDialogStd OmcDialogConsole OmcCompiler OmcHostDialog OmcConsole
OmlOPG OmlOPL OmlOPF OmlOPC OmlOPV OmlBackEnd OmlCompiler OmlConsole
```

2 Compiling Examples

2.1. Compiling examples for 32-bit:

```
^Q OmlCompiler.CompileThis +HostConLog OmtestHelloWorld: OmtestFormats:
OmtestDateTime: OmtestMkTraps: OmtestHeap:
```

Expected result in ~/Omtest/Clwe/ directory: OmtestHelloWorld.ll OmtestHelloWorld.bc OmtestFormats.ll OmtestFormats.bc OmtestDateTime.ll OmtestDateTime.bc OmtestMkTraps.ll OmtestMkTraps.bc OmtestHeap.ll OmtestHeap.bc

2.2. Compiling examples for 64-bit:

```
^Q OmlCompiler.CompileThis -bits 64 +HostConLog OmtestHelloWorld: OmtestFormats:
OmtestDateTime: OmtestMkTraps: OmtestHeap:
```

Expected result in ~/Omtest/Clwr/ directory: OmtestHelloWorld.ll OmtestHelloWorld.bc OmtestFormats.ll OmtestFormats.bc OmtestDateTime.ll OmtestDateTime.bc OmtestMkTraps.ll OmtestMkTraps.bc OmtestHeap.ll OmtestHeap.bc

3 Self-Compiling Shell

3.1 Self-Compile 32-bit LLVM Services.

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

3.2 Self-Compile 32-bit console Oberon Shell.

```
^Q OmlCompiler.CompileThis -System Syslwe -Host Hostlwe -options lb -directories
Llwe LlvmNative OmcTarget OmcCRuntime OmcDialog OmcHooks OmcTxtSource
OmcDialogConsole OmcRuntimeStd OmcOPM OmcOPT OmcOPB OmcOPU OmcOPS OmcOPP
```

```
OmcParams OmcConsole OmcDump OmcShell OmlOPG OmlOPL OmlOPF OmlOPC OmlOPV
OmlBackEnd OmlLoader OmlSh:
```

3.3 Self-Compile 64-bit LLVM Services.

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

3.4 Self-Compile 64-bit console Oberon Shell.

```
^Q OmlCompiler.CompileThis -System Syslwr -Host Hostlwr -options lb -directories
Llwr -bits 64 LlvmNative OmcTarget OmcCRuntime OmcDialog OmcHooks OmcTxtSource
OmcDialogConsole OmcRuntimeStd OmcOPM OmcOPT OmcOPB OmcOPU OmcOPS OmcOPP
OmcParams OmcConsole OmcDump OmcShell OmlOPG OmlOPL OmlOPF OmlOPC OmlOPV
OmlBackEnd OmlLoader OmlSh:
```

8 Unloading Oml Compiler

```
^Q DevDebug.UnloadThis OmlCompiler OmlBackEnd OmlOPV OmlOPC OmlOPF OmlOPL OmlOPG
OmcCompiler OmcDialogStd OmcRuntimeStd OmcOdcSource OmcParams OmcDump OmcOPP
OmcOPS OmcOPU OmcOPB OmcOPT OmcOPM OmcDialog OmcHooks OmcCRuntime OmcTarget
```

9 Change log

may 2019 original MultiOberon pre-version 0.8 released

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

Dmitry V. Dagaev
dvdagaev@yahoo.com