Building C++ Boost and Boosting C++ Builds

Vladimir Prus, Joom

Boost.Build

Система сборки для C++ с открытым кодом

Контекст: Building C++ Boost
Будущее: Boosting C++ Builds

Boost C++ Libraries, 2001

- Тестовая площадка для новых стандартных библиотек
- Переносимость - одно из главных требований

- Десять разных компиляторов
- Windows/Linux/Mac/Solaris/QNX
- Системы сборки это сложно

Req. №1: Декларативное описание

Что собирать

- Библиотеки и тесты
- Список исходных файлов

Как собирать

- Команды и опции конкретных компиляторов
- Разные наборы файлов и преобразований для разных платформ

Переносимые свойства сборки

Req. №2: Варианты сборки

- Разные компиляторы
- С оптимизацией и без
- Динамические и статические библиотеки

В любых комбинациях для любых частей проекта

Средства сборки в 2001

- automake/autoconf
- CMake
- SCons

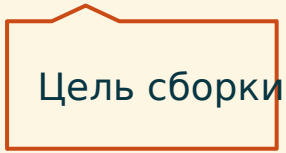
Boost.Build

http://boost.org/build

- V1: 2001: Proof of concept
  - Dave Abrahams, Rene Rivera, Vladimir Prus
- V2: 2005: Current version
  - Vladimir Prus, Steven Watanabe, Rene Revera
- V3: 2017: Python port, IDE support, incremental rebuilds

Hello, World

```
lib hello : hello.cpp ;
```




```
$ b2
gcc -o bin/gcc/libhello.so
```

Hello, static World

```
lib hello : hello.cpp ;
```

Свойство сборки

```
$ b2 link=static
ar bin/gcc/link-static/libhello.a
```

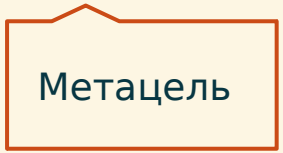
Hello, Windows World

```
lib hello : hello.cpp ;
```

```
$ b2 toolset=msvc
link bin/msvc/link-shared/hello.dll
```

Hello, both Worlds

```
lib hello : hello.cpp ;
```




```
$ b2 toolset=gcc toolset=msvc
gcc -o bin/gcc/libhello.so
link bin/msvc/hello.dll
```

Requirements

```
lib hello : hello.cpp : <optimization>space ;
```

```
$ b2
gcc -o bin/gcc/optimization-space/libhello.so
```

Conditional Requirements 1

```
if toolset == msvc {
   lib hello : hello.cpp <link>static ;
} else {
   lib hello : hello.cpp ;
}
```

Conditional Requirements

```
lib hello : hello.cpp : <toolset>msvc:<link>static;


rule customize ( properties * ) {
    if <toolset>msvc in $(properties) {
        return <link>static ;
}
lib hello2 : hello.cpp : <conditional>@customize ;
```

Project Requirements

```
project boost
    : requirements <include>$(BOOST_ROOT)
    ;

lib hello : hello.cpp ;
```

Usage Requirements

```
project boost
    : requirements <include>$(BOOST_ROOT)
    : usage-requirements <include>$(BOOST_ROOT)
    ;

lib hello : hello.cpp ;

exe app : app.cpp /boost//hello ;
```

Сборка библиотеки Boost C++

```
boost-lib program_options
    : $(SOURCES).cpp
    : <target-os>hpux,<toolset>gcc:↲
      <define>_INCLUDE_STDC__SOURCE_199901
    ;
```

Тестирование библиотеки

```
project
    : requirements
     <library>../build//boost_program_options
    ;

test-suite program_options :
    [ run parsers_test.cpp ]
    [ run parsers_test.cpp : <link>shared ]
    ;
```

Использование PCH

```
cpp-pch pch : pch.hpp ;
run test_gamma_dist.cpp pch ;
```

Плюсы и минусы

- Простое описание отдельных проектов
- Поддержка вариантов сборки
- Переносимость

- Документация
- “Слишком много магии”
- Интеграция с IDE

Boosting C++ Builds

- Осталось 3 основных компилятора
  - Но существующие постоянно меняются
- Много платформ
  - Android, iOS, Windows RT, Embedded
- Все те же C++ разработчики

Проблемы все те же

Другие системы: CMake

```
add_library(foo ...)
set_property(
    TARGET foo
    PROPERTY INTERFACE_INCLUDE_DIRECTORIES
"/opt/foo/include")
```

Другие системы: QBS

```
Library {
    cpp.optimization: “small”
    Properties {
        condition: qbs.buildVariant == "debug"
        cpp.defines: ["QS_LOG_LINE_NUMBERS"]
    }
}
```

Другие системы: QBS

```
Library {
    Export {
        cpp.includePaths: [product.sourceDirectory]
     }
}
```

Boost.Build 2016

- Независима от Boost C++ Libraries
- Простая инсталляция
- Python версия
- http://boost.org/build

Спасибо за внимание

- vladimir.prus@gmail.com
- @vladimirprus