ClangのModules

Cocoa勉強会 #62 / 2013-10-19 / 木村渡

Powered by Rabbit 2.1.1 and COZMIXNG

Highlights of Xcode 5

Compiler

- **Modules** for system frameworks to speed build time
- Auto Link frameworks imported by **code modules** (...)

"What's New in Xcode"

Modules

- Xcode 5 (Clang 3.3)の新機能
- iOS 7 or Mac OS X 10.9
- C, Objective-C (C++ not ready)

ぶっちゃけ、あんまり
気にしなくてよい

Xcodeがよきに
はからってくれる

Modules

コンパイルが速くなる(らしい)

- ある種のprecompiled header
- module単位でバイナリを生成
  - ASTをシリアライズしたもの(らしい)

fragileがうんぬんは他の資料を見てね。

Modules @import

```
// including header file
#import <Foundation/Foundation.h>
#import <Foundation/NSString.h>

// using Modules
@import Foundation;
@import Foundation.NSString;
```

あっとまーく + import。
ヘッダファイル名でなくモジュール名を書く。
通常ヘッダファイル名とモジュール名は同じ。

Modules Auto Link

コード中に@import

→必要なものを**自動的に**リンク。

リンクするライブラリやフレームワークの指示が不要に。

Modules Auto Linking

clangを直接たたいてるなら、リンカオプションが不要に。

```
-llib
-framework f
```

標準の場所にないとき、-Iや-Fは必要

話すこと

- modulesとは（済み）
- module.map
- 試してみる
- clang-3.4での新機能（の予定）

module.map

モジュールを定義するファイル。

これさえあればライブラリ、フレームワークがモジュール対応に。

module.mapの中身はテキスト

module.map

```
$ cd `xcrun --sdk iphoneos
            --show-sdk-path`
$ find . -name module.map
    :
./System/Library/Frameworks/
          Foundation.framework/module.m
    :
./System/Library/Frameworks/
          StoreKit/module.map
    :
```

module.map

```
./System/Library/Frameworks/
          UIKit.framework/module.map
    :
./usr/include/module.map
./usr/include/objc/module.map
```

10.9は未リリースだから、iOS7のSDKにあるmodule.mapを見てみよう。
フレームワークとinludeとヘッダファイルがあったような場所にmodule.mapが置かれている。

Foundation.framework/module.map

```
framework module Foundation [system] {
  umbrella header "Foundation.h"

  export *
  module * {
    export *
  }
```

module.mapの中身。詳しくはClangのドキュメント見てね。

/usr/include/module.map

```
module Darwin [system] {
  // Headers that are repeatedly inclu
  // assigned to any given module.
  exclude header "_structs.h"
  exclude header "sys/_structs.h"

  module cdefs {
    header "sys/cdefs.h"
    export *
  }
```

/usr/include/module.map

```
// C standard library
module C {
  module complex {
    header "complex.h"
    export *
  }
  :
```

"Module Map Language"

まだまだ変わりそう。clang 3.4で予約語も増えてる。

iOS7 SDKでは使われてない機能も。

```
config_macros
conflict
link
```

モジュールとライブラリの名前ちがうときにlink使うぽい。

"Module declaration"

```
Name.framework/
   module.map    /* module map file */
   Headers/
   Frameworks/
   Resources/
   Name
```

.frameworkのすぐ下にmodule.mapを置くようになっている。

使ってみる

iOS 7.0 SDKを参考に、Mountain Lionにmodule.mapをつっこむ！

```
framework module Foundation [system] {
  umbrella header "Foundation.h"

  export *
  module * {
    export *
  }
}
```

mod1.m / 今までのやり方

```
#import <Foundation/Foundation.h>
int main (int argc, const char * argv
  @autoreleasepool {
    NSLog(@"Hello, modules!");
  }
  return 0;
}
```

mod1.m / 今までのやり方

```
% clang mod1.m -o mod1
        -framework Foundation
  # framework name ^^^^^^^^^^
% otool -L mod1
mod1:
  /System/Library/Frameworks/
          Foundation.framework/
          Versions/C/Foundation (...)
  :
```

mod2.m / Modulesを使う

```
@import Foundation;
/* ^^^^ using modules */
int main (int argc, const char * argv
  @autoreleasepool {
    NSLog(@"Hello, modules!");
  }
  return 0;
}
```

mod2.m / Modulesを使う

```
% clang mod2.m -o mod2 -fmodules
      # no -framework   ^^^^^^^^^
% otool -L mod2
mod2:
   :
   /System/Library/Frameworks/
            Foundation.framework/
            Versions/C/Foundation (...)
```

#importを@importに読み替え

```
% clang mod1.m -o mod1 -fmodules
% ./mod1
2013-10-18 ...[] Hello, modules!
# works fine!
```

ソースコード中は#importなのにモジュールが使えてる。
リンクするフレームワークを-frameworkで指定してない→自動リンクが機能してる。

autolinkを無効に

```
% clang mod1.m -o mod1 -fmodules
                      -fno-autolink
  #   disable autolink ^^^^^^^^^^^^^
Undefined symbols for architecture x8
  "_NSLog", referenced from:
      _main in mod1-dTLx89.o
  :
```

Xcodeでの設定

- CLANG_ENABLE_MODULES:
  - YES → -fmodules
- CLANG_MODULES_AUTOLINK:
  - NO → -fno-autolink

ここまでのまとめ

- module.mapがあればmodulesが使える
- #importも-fmodulesで@importとして解釈される
  - →コード修正不要
  - Apple版でない、clang-3.3でも同様

mod3.m / cache

```
@import Foundation.NSObjCRuntime;
/* ^^^^ using modules */
int main (int argc, const char * argv
  @autoreleasepool {
    NSLog(@"Hello, modules!");
  }
  return 0;
}
```

mod3.m / cache

```
% clang mod3.m -o mod3 -fmodules
        -fmodules-cache-path=./tmp
% find ./tmp
./tmp
./tmp/B76TNZTEB88L
./tmp/B76TNZTEB88L/_Builtin_intrinsic
./tmp/B76TNZTEB88L/Foundation.pcm
./tmp/B76TNZTEB88L/modules.idx
./tmp/modules.timestamp
```

mod3.m / cache

```
% clang -module-file-info
  ./tmp/.../Foundation.pcm
Information for module file
 './tmp/.../Foundation.pcm':
  Generated by this Clang: (clang-500
  Language options:
    C99: Yes
    C11: No
    Microsoft extensions: No
  :
```

mod3.m / cache

```
% strings ./tmp/.../Foundation.pcm |
  grep ^NS
NS,I
NSs@
NSGreaterThanPredicateOperatorType
NSURLRequest
NSInvalidArchiveOperationException
  :
```

シンボルも入ってるぽい。

Problems modules do not solve

- Rewrite the world's code
- Versioning
- Namespaces
- Binary distribution of modules

"Clang 3.3 documentation"

clang-3.4 modules

- C++ support (experimental)
- compiler options
  - -fmodule-maps, -fmodule-map-file, ...
- module map lang
  - private, extern, use

clang-3.4 modularize

- clang-tools-extra
- generate module.map (r192703)

ヘッダファイルからmodule.mapを生成する機能が追加された。

Resources

- Clang documentation
- WWDC 2013

感想

- ユーザ(開発者)があまり気を回さなくてもうまくいくよう、よく準備されてる
- まだまだ変わりそう
- ヘッダごとでなくmodule単位で生成することでパフォーマンス稼いでる？

終