链滴

如何快速构建一个简单的c/c++程序

作者：waruqi
原文链接：https://ld246.com/article/1468681805392
来源网站：链滴

首先我们通过内置的工程模板创建一个空工程:

```
$ xmake create -P ./hello

create hello ...
create ok!:ok_hand:
```

这个时候xmake将会产生一些工程文件，如下:

```
$ cd ./hello
$ tree .

.
├── src
│   └── main.c
└── xmake.lua
```

这个简单的程序仅仅只是为了打印输出： hello xmake!

```
$ cat ./src/main.c

#include <stdio.h>
int main(int argc, char** argv)
{
    printf("hello xmake!\n");
    return 0;
}
```

xmake.lua是基于lua语法的工程描述文件，它很简单:

```
$ cat xmake.lua

target("hello")
    set_kind("binary")
    add_files("src/*.c")
```

现在我们开始编译这个程序

```
$ xmake

checking for the architecture ... x86_64
checking for the Xcode SDK version for macosx ... 10.11
checking for the target minimal version ... 10.11
checking for the c compiler (cc) ... xcrun -sdk macosx clang
checking for the c++ compiler (cxx) ... xcrun -sdk macosx clang
checking for the objc compiler (mm) ... xcrun -sdk macosx clang
checking for the objc++ compiler (mxx) ... xcrun -sdk macosx clang++
checking for the assember (as) ... xcrun -sdk macosx clang
checking for the linker (ld) ... xcrun -sdk macosx clang++
checking for the static library archiver (ar) ... xcrun -sdk macosx ar
checking for the static library extractor (ex) ... xcrun -sdk macosx ar
checking for the shared library linker (sh) ... xcrun -sdk macosx clang++
checking for the swift compiler (sc) ... xcrun -sdk macosx swiftc
```

```
checking for the debugger (dd) ... xcrun -sdk macosx lldb
configure
{
    ex = "xcrun -sdk macosx ar"
,   ccache = "ccache"
,   plat = "macosx"
,   ar = "xcrun -sdk macosx ar"
,   buildir = "build"
,   as = "xcrun -sdk macosx clang"
,   sh = "xcrun -sdk macosx clang++"
,   arch = "x86_64"
,   mxx = "xcrun -sdk macosx clang++"
,   xcode_dir = "/Applications/Xcode.app"
,   target_minver = "10.11"
,   sc = "xcrun -sdk macosx swiftc"
,   mode = "release"
,   make = "make"
,   cc = "xcrun -sdk macosx clang"
,   host = "macosx"
,   dd = "xcrun -sdk macosx lldb"
,   kind = "static"
,   ld = "xcrun -sdk macosx clang++"
,   xcode_sdkver = "10.11"
,   cxx = "xcrun -sdk macosx clang"
,   mm = "xcrun -sdk macosx clang"
}
configure ok!
clean ok!
[00%]: ccache compiling.release src/main.c
[100%]: linking.release hello
build ok!:ok_hand:
```

接着运行它:

```
$ xmake run hello

hello world!
```

或者进行调试

```
$ xmake run -d hello

[lldb]$target create "build/hello"
Current executable set to 'build/hello' (x86_64).
[lldb]$b main
Breakpoint 1: where = hello`main, address = 0x0000000100000f50
[lldb]$r
Process 7509 launched: '/private/tmp/hello/build/hello' (x86_64)
Process 7509 stopped
* thread #1: tid = 0x435a2, 0x0000000100000f50 hello`main, queue = 'com.apple.main-thread
, stop reason = breakpoint 1.1
    frame #0: 0x0000000100000f50 hello`main
hello`main:
->  0x100000f50 <+0>:  pushq  %rbp
```

```
    0x100000f51 <+1>:  movq   %rsp, %rbp
    0x100000f54 <+4>:  leaq   0x2b(%rip), %rdi          ; "hello world!"
    0x100000f5b <+11>: callq  0x100000f64               ; symbol stub for: puts
[lldb]$
```

接着我们尝试构建一个android版本，这个时候得设置ndk路径，当然也能配置到全局配置中，一劳逸

```
$ xmake f -p android --ndk=~/files/android-ndk-r10e/

checking for the architecture ... armv7-a
checking for the SDK version of NDK ... android-21
checking for the c compiler (cc) ... arm-linux-androideabi-gcc
checking for the c++ compiler (cxx) ... arm-linux-androideabi-g++
checking for the assember (as) ... arm-linux-androideabi-gcc
checking for the linker (ld) ... arm-linux-androideabi-g++
checking for the static library archiver (ar) ... arm-linux-androideabi-ar
checking for the static library extractor (ex) ... arm-linux-androideabi-ar
checking for the shared library linker (sh) ... arm-linux-androideabi-g++
configure
{
    ex = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/prebuilt/da
win-x86_64/bin/arm-linux-androideabi-ar"
,   ccache = "ccache"
,   ndk = "~/files/android-ndk-r10e/"
,   sc = "xcrun -sdk macosx swiftc"
,   ar = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/prebuilt/da
win-x86_64/bin/arm-linux-androideabi-ar"
,   ld = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/prebuilt/da
win-x86_64/bin/arm-linux-androideabi-g++"
,   buildir = "build"
,   host = "macosx"
,   as = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/prebuilt/da
win-x86_64/bin/arm-linux-androideabi-gcc"
,   toolchains = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/pr
built/darwin-x86_64/bin"
,   arch = "armv7-a"
,   mxx = "xcrun -sdk macosx clang++"
,   xcode_dir = "/Applications/Xcode.app"
,   target_minver = "10.11"
,   ndk_sdkver = 21
,   mode = "release"
,   cc = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/prebuilt/da
win-x86_64/bin/arm-linux-androideabi-gcc"
,   cxx = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/prebuilt/d
rwin-x86_64/bin/arm-linux-androideabi-g++"
,   make = "make"
,   dd = "xcrun -sdk macosx lldb"
,   kind = "static"
,   sh = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/prebuilt/da
win-x86_64/bin/arm-linux-androideabi-g++"
,   xcode_sdkver = "10.11"
,   plat = "android"
,   mm = "xcrun -sdk macosx clang"
```

```
}
configure ok!

$ xmake

clean ok!
[00%]: ccache compiling.release src/main.c
[100%]: linking.release hello
build ok!:ok_hand:
```

或者我们编一个iphoneos的版本，例如：

```
$ xmake f -p iphoneos

checking for the architecture ... armv7
checking for the Xcode SDK version for iphoneos ... 9.2
checking for the target minimal version ... 9.2
checking for the c compiler (cc) ... xcrun -sdk iphoneos clang
checking for the c++ compiler (cxx) ... xcrun -sdk iphoneos clang
checking for the objc compiler (mm) ... xcrun -sdk iphoneos clang
checking for the objc++ compiler (mxx) ... xcrun -sdk iphoneos clang++
checking for the assember (as) ... gas-preprocessor.pl xcrun -sdk iphoneos clang
checking for the linker (ld) ... xcrun -sdk iphoneos clang++
checking for the static library archiver (ar) ... xcrun -sdk iphoneos ar
checking for the static library extractor (ex) ... xcrun -sdk iphoneos ar
checking for the shared library linker (sh) ... xcrun -sdk iphoneos clang++
checking for the swift compiler (sc) ... xcrun -sdk iphoneos swiftc
configure
{
    ex = "xcrun -sdk iphoneos ar"
,   ccache = "ccache"
,   ndk = "~/files/android-ndk-r10e/"
,   sc = "xcrun -sdk iphoneos swiftc"
,   ar = "xcrun -sdk iphoneos ar"
,   sh = "xcrun -sdk iphoneos clang++"
,   buildir = "build"
,   xcode_dir = "/Applications/Xcode.app"
,   as = "/usr/local/share/xmake/tools/utils/gas-preprocessor.pl xcrun -sdk iphoneos clang"
,   toolchains = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/pr
built/darwin-x86_64/bin"
,   arch = "armv7"
,   mxx = "xcrun -sdk iphoneos clang++"
,   ndk_sdkver = 21
,   target_minver = "9.2"
,   cc = "xcrun -sdk iphoneos clang"
,   mode = "release"
,   host = "macosx"
,   cxx = "xcrun -sdk iphoneos clang"
,   make = "make"
,   dd = "xcrun -sdk macosx lldb"
,   kind = "static"
,   ld = "xcrun -sdk iphoneos clang++"
,   xcode_sdkver = "9.2"
,   plat = "iphoneos"
```

```
,   mm = "xcrun -sdk iphoneos clang"
}
configure ok!

$ xmake

[00%]: ccache compiling.release src/main.c
[100%]: linking.release hello
build ok!:ok_hand:
```

最后我们尝试为mingw平台进行编译，sdk指定交叉工具链目录，交叉编译linux平台也可以这么用哦。

```
$ xmake f -p mingw --sdk=/usr/local/i386-mingw32-4.3.0/

checking for the architecture ... i386
checking for the c compiler (cc) ... i386-mingw32-gcc
checking for the c++ compiler (cxx) ... i386-mingw32-g++
checking for the assember (as) ... i386-mingw32-gcc
checking for the linker (ld) ... i386-mingw32-g++
checking for the static library archiver (ar) ... i386-mingw32-ar
checking for the static library extractor (ex) ... i386-mingw32-ar
checking for the shared library linker (sh) ... i386-mingw32-g++
checking for the swift compiler (sc) ... no
configure
{
    ex = "/usr/local/i386-mingw32-4.3.0/bin/i386-mingw32-ar"
,   ccache = "ccache"
,   ndk = "~/files/android-ndk-r10e/"
,   sc = "xcrun -sdk iphoneos swiftc"
,   sdk = "/usr/local/i386-mingw32-4.3.0/"
,   cc = "/usr/local/i386-mingw32-4.3.0/bin/i386-mingw32-gcc"
,   ndk_sdkver = 21
,   buildir = "build"
,   plat = "mingw"
,   as = "/usr/local/i386-mingw32-4.3.0/bin/i386-mingw32-gcc"
,   toolchains = "/Users/ruki/files/android-ndk-r10e/toolchains/arm-linux-androideabi-4.9/pr
built/darwin-x86_64/bin"
,   arch = "i386"
,   mxx = "xcrun -sdk iphoneos clang++"
,   xcode_dir = "/Applications/Xcode.app"
,   target_minver = "9.2"
,   sh = "/usr/local/i386-mingw32-4.3.0/bin/i386-mingw32-g++"
,   mode = "release"
,   host = "macosx"
,   cxx = "/usr/local/i386-mingw32-4.3.0/bin/i386-mingw32-g++"
,   make = "make"
,   dd = "xcrun -sdk macosx lldb"
,   kind = "static"
,   ar = "/usr/local/i386-mingw32-4.3.0/bin/i386-mingw32-ar"
,   xcode_sdkver = "9.2"
,   ld = "/usr/local/i386-mingw32-4.3.0/bin/i386-mingw32-g++"
,   mm = "xcrun -sdk iphoneos clang"
}
```

```
configure ok!

$ xmake

[00%]: ccache compiling.release src/main.c
[100%]: linking.release hello.exe
build ok!:ok_hand:
```

xmake还能直接在windows的cmd终端下，进行直接编译windows的程序，它会去自动检测当前系装的vs环境，调用里面的cl.exe编译器进行编译，一切都是自动化的，我们不需要额外配置什么，只要执行：`xmake` 就行了。。

例如：

```
$ xmake

checking for the architecture ... x86
checking for the Microsoft Visual Studio version ... 2008
checking for the c compiler (cc) ... cl.exe
checking for the c++ compiler (cxx) ... cl.exe
checking for the assember (as) ... ml.exe
checking for the linker (ld) ... link.exe
checking for the static library archiver (ar) ... link.exe -lib
checking for the shared library linker (sh) ... link.exe -dll
checking for the static library extractor (ex) ... lib.exe
configure
{
    ex = "lib.exe"
,   sh = "link.exe -dll"
,   host = "windows"
,   ar = "link.exe -lib"
,   as = "ml.exe"
,   plat = "windows"
,   buildir = "build"
,   arch = "x86"
,   cc = "cl.exe"
,   cxx = "cl.exe"
,   mode = "release"
,   clean = true
,   kind = "static"
,   ld = "link.exe"
,   vs = "2008"
}
configure ok!
[00%]: compiling.release src\main.c
[100%]: linking.release hello.exe
build ok!
```

顺便说一下，在windows下编译，xmake是完全支持多任务的哦，默认就是自动多任务构建的，比起前在msys, cygwin里面用 gmake来编译快多了，因为windows下的gmake就算你启用了`-j 4` 也没啥果，非常非常得慢。。。

- XMake项目主页

- XMake项目详情
- XMake项目源码
- XMake项目文档