链滴

xmake v2.3.8 发布, 新增 Intel C++/Fortran 编译器支持

作者：waruqi

原文链接：https://ld246.com/article/1602985363389

来源网站：链滴

xmake 是一个基于 Lua 的轻量级跨平台构建工具，使用 xmake.lua 维护项目构建，相比 makefile/MakeLists.txt，配置语法更加简洁直观，对新手非常友好，短时间内就能快速入门，能够让用户把更的精力集中在实际的项目开发上。

在这个新版本中，我们对 Intel 系列的 C++ 和 Fortran 编译器做了全平台支持，并且改进了上个版新加的 Wasm 工具链支持，同时对 Qt SDK for Wasm 也进行了支持。

另外，我们还将 luajit 升级到最新的 v2.1 版本，在跨平台方面，xmake 也做了很大的完善，增加了 mps64 架构的运行支持。

- 项目源码
- 官方文档
- 入门课程

新特性介绍

Intel C++ 编译器支持

这个版本上，我们对 Intel 系列的 C++ 编译器做了全平台支持，包括 windows上的 icl 以及 linux/mcOS 下的 icc/icpc。

要想启用 Intel C++ 编译器，我们只需要在安装了的 Intel 编译器的系统上，通过 --toolchain=icc 数切换到对应的工具链即可。

```
$ xmake f --toolchain=icc
$ xmake
```

Intel Fortran 编译器支持

之前的版本，xmake 仅仅支持 gfortran 编译器，而这个版本，我们也支持了 Intel Fortran 编译器也就是 ifort，我们也只需要切换到对应的 ifort 工具链即可使用。

```
$ xmake f --toolchain=ifort
$ xmake
```

Wasm 平台和 Qt/Wasm 支持

上个版本，我们新增了 --toolchain=emcc 工具链来支持 wasm 程序的编译，但是仅仅指定工具链并不能很好的调整目标程序的扩展名，例如对 *.js 和 *.wasm 的文件生成。

而新版本，我们继续新增了 xmake f -p wasm 平台，内置启用 emcc 工具链，并且再次基础上对周的配置做了更好的完善。

只要切换到 wasm 平台，xmake 会默认生成 *.js 以及对应 *.wasm 等目标文件，还会额外生成可以载 js 来运行 wasm 程序的 *.html 页面。

另外，我们还对 Qt SDK for Wasm 也做了支持，比如我们创建一个 Qt QuickApp 工程。

```
$ xmake create -t qt.quickapp_static quickapp
```

这里，我们注意到，我们创建的是需要静态link的Qt工程，因为 wasm 版本的 Qt 库，我们需要强制

态链接到程序才能正常使用。

生成的工程文件 xmake.lua 内容大概如下：

```
add_rules("mode.debug", "mode.release")

includes("qt_add_static_plugins.lua")

target("demo")
    add_rules("qt.quickapp_static")
    add_headerfiles("src/*.h")
    add_files("src/*.cpp")
    add_files("src/qml.qrc")
    add_frameworks("QtQuickControls2", "QtQuickTemplates2")
    qt_add_static_plugins("QtQuick2Plugin", {linkdirs = "qml/QtQuick.2", links = "qtquick2plugi
"})
    qt_add_static_plugins("QtQuick2WindowPlugin", {linkdirs = "qml/QtQuick/Window.2", links
= "windowplugin"})
    qt_add_static_plugins("QtQuickControls2Plugin", {linkdirs = "qml/QtQuick/Controls.2", links
= "qtquickcontrols2plugin"})
    qt_add_static_plugins("QtQuickTemplates2Plugin", {linkdirs = "qml/QtQuick/Templates.2", l
nks = "qtquicktemplates2plugin"})
```

上面的配置中，我们除了启用 qt.quickapp_static 编译规则，还通过 qt_add_static_plugins 配置了些必须的 Qt 插件。

接下来，我们只需要切换到 wasm 平台，并确保 Qt SDK 已设置，即可完成编译。

```
$ xmake f -p wasm [--qt=~/Qt]
$ xmake
```

完成编译后，xmake 会在 build 目录下生成 demo.html 以及对应的 demo.js/demo.wasm 程序，们打开 demo.html 页面即可运行我们编译的 Qt 程序，显示效果如下图：

关于 Qt/Wasm 更加详细的说明，见：Issue #956

新增 Math/Float-point 编译优化设置

我们新增了一个 set_fpmodels() 设置接口，用于设置浮点的编译模式，对数学计算相关优化的编译

象设置，提供：fast, strict, except, precise 等几种常用的级别，有些可同时设置，有些是有冲突的最后设置的生效。

关于这些级别的说明，可以参考下微软的文档：Specify floating-point behavior

当然，对应gcc/icc等其他编译器，xmake 会映射到不同的编译flags。

```
set_fpmodels("fast")
set_fpmodels("strict")
set_fpmodels("fast", "except")
set_fpmodels("precise") -- default
```

关于这块详情见：Issue #981

OpenMP 支持

为了更加抽象简单的启用 openmp 特性，我们可以通过新增的 c.openmp 和 c++.openmp 这两个则来设置，另外 linux、macOS 上我们需要额外的 libomp 库才行，因此可以通过 add_requires("libomp") 来快速引用和集成。

```
add_requires("libomp", {optional = true})
target("loop")
    set_kind("binary")
    add_files("src/*.cpp")
    add_rules("c++.openmp")
    add_packages("libomp")
```

如果是c代码，需要启用 add_rules("c.openmp")，如果是 c/c++ 混合编译，那么这两个规则都要设。

c11/c17 的支持

新版本中，xmake 对 set_languages 也做了改进，增加了新的 c11/c17 设置项，同时对最新版本 msc 提供的 /std:c11 和 /std:c17 也做了适配和支持。

我们只需要简单的设置：

```
set_languages("c17")
```

即可启用 c17 标准来编译，即使低版本 msvc 等编译器不支持，xmake 也会自动忽略设置。

更好的 Mingw 支持

关于这块的改进，涉及几个方面，首先是 Windows 下 Mingw SDK 根目录的自动探测的改进，大部情况下，我们都不需要额外配置 --mingw= 参数显式指定路径，也能够自动检测到了。

关于这块详情，见：Issue #977

另外，除了 Msys2/Mingw 还有 macOS,linux/Mingw，我们在新版本中还额外支持了 llvm-mingw 这个 SDK，使得我们可以使用 mingw 来编译 arm/arm64 架构的程序。

```
$ xmake f -p mingw -a arm64
$ xmake
```

另外，在远程依赖包的自动编译集成上，现在带有 cmakelists 的第三方库，即使是 mingw 平台，xmake 也能自动编译集成进来直接使用，非常的快速方便。

而最近 xmake-repo 官方 C/C++ 包仓库中，我们也新增收入了不少支持 mingw 平台的新库，可直使用。

更好的跨平台运行

我们新增了对 mips64 架构的 linux 系统运行支持，另外改进了 arm/arm64 下 xmake 的运行稳定，通过合入最新的 luajit v2.1 ，解决了很多 luajit 遗留的问题，比如 arm64 下 lightuserdata 的 bad pointer 等问题。

新增 macOS Sierra for arm64 支持

xmake 还对最新的 Xcode-beta 进行了适配，新增了 macOs for arm64 的目标程序编译支持，只要切到 arm64 架构编译即可。

```
$ xmake f -a arm64 [--xcode=Applications/Xcode-beta.app/]
$ xmake
```

当然，前提是在 macOS 下运行，并且使用最新支持 Developer Transition Kit (DTK) 的 Xcode-beta 版本才行。

官方仓库收录更多的 C/C++ 库

在 xmake 的官方 C/C++ 仓库 xmake-repo 中，我们最近新增了几十个常用的 C/C++ 库，并且还对 ibx11 系列的库都全部进行了收录。

虽然，仓库的包维护工作量巨大，但是目前的发展趋势也日趋活跃，我们收到了越来越多的用户对仓包的贡献和改进维护。

并且，现在我们的官方仓库已经可以快速集成：linux, macOS, windows, mingw, bsd, msys, iphonos, android 等八大常用平台的库，实现真正的跨平台 C/C++ 远程依赖库集成和使用支持。

bsd	linux	windows	macosx	msys	mingw	iphoneos	android
bin2c	abseil	abseil	abseil	bin2c	bin2c	bin2c	bin2c
catch2	assimp	assimp	assimp	catch2	box2d	catch2	catch2
concurrentqueue	bin2c	bin2c	autoconf	concurrentqueue	catch2	cjson	cjson
cpp-taskflow	boost	boost	automake	cpp-taskflow	concurrentqueue	concurrentqueue	concurrentqueue
cxxopts	box2d	box2d	bin2c	cxxopts	cpp-taskflow	cpp-taskflow	cpp-taskflow
doctest	bullet3	bullet3	boost	doctest	cxxopts	cxxopts	cxxopts
fmt	bzip2	bzip2	box2d	fmt	doctest	doctest	doctest
gtest	cairo	cairo	bullet3	gtest	fmt	fmt	ffmpeg
inja	catch2	catch2	bzip2	inja	glew	gtest	fmt
irrXML	cjson	co	cairo	irrXML	glfw	imgui	gtest
libjpeg	co	concurrentqueue	catch2	libjpeg	gtest	inja	imgui
lua	concurrentqueue	cpp-taskflow	cjson	libxmake	imgui	irrXML	inja
luajit	cpp-taskflow	cxxopts	cmake	nlohmann_json	inja	json-c	irrXML
moonjit	cxxopts	doctest	co	pybind11	irrXML	libcurl	json-c
ncurses	doctest	eigen	concurrentqueue	spdlog	libjpeg	libev	libjpeg
nlohmann_json	eigen	expat	cpp-taskflow	stb	libraw	libffi	libpng
pybind11	expat	fmt	cxxopts	tbox	libsdl	libjpeg	libuv
spdlog	expresscpp	freeglut	doctest	tmxparser	libsdl_image	libpng	libxmake
stb	ffmpeg	freetype	eigen		libsdl_mixer	libraw	libxml2
tbox	fmt	glew	expat		libsdl_net	libuv	lua
tmxparser	fontconfig	glfw	expresscpp		libsdl_ttf	libxml2	luajit

目前我们收录的一些包列表和支持平台，可以从这里查看：PKGLIST.md

我们一直在努力解决 C/C++ 库生态的杂乱、集成使用繁琐等问题，提供快速一致的自动集成和编译案，xmake 不仅支持 vcpkg/conan/clib/homebrew 等第三方官方仓库包的集成，并且也在努力完自建的官方仓库，实现更好的集成体验。

例如：

```
add_requires("tbox >1.6.1", "libuv master", "vcpkg::ffmpeg", "brew::pcre2/libpcre2-8")
add_requires("conan::openssl/1.1.1g", {alias = "openssl", optional = true, debug = true})
target("test")
    set_kind("binary")
    add_files("src/*.c")
    add_packages("tbox", "libuv", "vcpkg::ffmpeg", "brew::pcre2/libpcre2-8", "openssl")
```

带有 vcpkg::，brew:: 和 conan:: 等命名空间的包，会自动切换到对应的第三方包仓库去下载集成，默认的 tbox >1.6.1 等库，则会默认使用 xmake-repo 官方仓库中提供的包。

使用和集成方式完全一致，xmake 会自动下载、编译、集成和链接。

关于远程包的依赖集成的更多详细说明，我们可以看下相关的文档说明：远程依赖库集成和使用

同时，我们也欢迎更多的人参与进来，帮忙一起改善 C/C++ 的库生态的建设，提供简洁一致的库使体验，我相信 C/C++ 包管理和库生态并不比 Rust/Go 差。

更多的发行版安装支持

在新版本中，我们将 xmake 提交了 Ubuntu PPA 源，因此除了现有的脚本安装方式外，我们也可以过 apt 去快速安装 xmake。

```
sudo add-apt-repository ppa:xmake-io/xmake
sudo apt update
sudo apt install xmake
```

同时，我们也将包提交到了 Copr 包管理仓库，使得我们也可以在 Fedora, RHEL，OpenSUSE，CentOS 等发行版中，通过 dnf 来快速安装 xmake。

```
sudo dnf copr enable waruqi/xmake
sudo dnf install xmake
```

入门课程

近期，我们也上线了官方的 xmake 入门课程，Xmake 带你轻松构建 C/C++ 项目 以边学边做实验方式快速学习 xmake 的使用。

更新内容

新特性

- #955: 添加 Zig 空工程模板
- #956: 添加 Wasm 编译平台，并且支持 Qt/Wasm SDK
- 升级luajit到v2.1最新分支版本，并且支持mips64上运行xmake
- #972: 添加depend.on_changed()去简化依赖文件的处理
- #981: 添加set_fpmodels()去抽象化设置math/float-point编译优化模式
- #980: 添加对 Intel C/C++ 和 Fortran 编译器的全平台支持
- #986: 对16.8以上msvc编译器增加c11/c17 支持
- #979: 添加对OpenMP的跨平台抽象配置。add_rules("c++.openmp")

改进

- #958: 改进mingw平台，增加对 llvm-mingw 工具链的支持，以及 arm64/arm 架构的支持
- 增加 add_requires("zlib~xxx") 模式使得能够支持同时安装带有多种配置的同一个包，作为独立存在
- #977: 改进 find_mingw 在 windows 上的探测
- #978: 改进工具链的flags顺序
- 改进XCode工具链，支持macOS/arm64

Bugs修复

- #951: 修复 emcc (WebAssembly) 工具链在windows上的支持
- #992: 修复文件锁偶尔打开失败问题