Vcpkg轻松集成开源第三方库

[[TOC]]

1. 为什么要用Vcpkg

1.1. 传统使用开源库的方法

Windows下开发C/C++程序，少不了编译开源的第三方库。比如用于网络连接的高性能库libcurl、用于压缩解压的zlib等等。使用这些库开发极大的方便了程序员，使得我们不必重复造轮子。但是使用这些库必须要处理以下问题。

1.1.1. 编译工具

由于这些开源库绝大部分都来源于Linux系统，导致其工程文件、编译系统都使用gnu系列工具，使得将其移植到Windows的VC开发环境下一直是难点。尽管最近几年很多开源库都支持了跨平台的cmake，但是编译过程仍然复杂和多样化。

常见的编译方式有：

编译方式	特点	举例
configure、make	需要msys这样的unix环境才可以编译	ffmpeg
自定义编译工具	需要学习特定的编译命令和工具	openssl、boost
cmake	相对简单轻松	libcurl
VC工程文件	这种最简单，直接拿来即可编译	

1.1.2. 编译类型

当了解了这些还不够，我们还需要考虑预先编译出哪种类型的开源库程序。比如：Debug还是Release、动态库还是静态库、MD还是MT、32位还是64位。光是这三种组合就有16种可能性。如果像libcurl这种还要考虑是否引用其他开源库的功能，那么编译类型的组合会更多。管理起来很麻烦。

1.1.3. 工程目录设定

由于多样的编译类型，工程目录也必须仔细设定才能保证自己的软件项目能够正常编译。

1.2. Vcpkg的优点

- 自动下载开源库源代码
- 源码包的缓存管理和版本管理，可以升级版本
- 轻松编译
- 依赖关系检查（比如编译libcurl，会自动下载zlib、openssl进行编译）
- 无缝集成Visual Studio，不需要设置库文件、头文件的所在目录，自动集成。
- Visual Studio全平台支持，不仅支持Debug/Release、x86/x64编译，还支持UWP、ARM平台的编译。

2. 获取Vcpkg

2.1. 下载Vcpkg

Vcpkg的官方源码站点为：

```
https://github.com/microsoft/vcpkg
```

一般地，你可以使用git命令克隆一个当前版本下来，或者直接下载压缩包。

```
git clone https://github.com/microsoft/vcpkg
```

2.2. 编译Vcpkg

注意：

> Vcpkg大量使用的psl脚本，所以官方强烈推荐使用PowerShell而不时CMD命令行来执行各种操作。尽管在使用的时候兼容CMD，但是在编译这一步，请使用PowerShell。

编译很简单，使用PowerShell执行Vcpkg工程目录下的“bootstrap-vcpkg.bat”命令，即可编译。编译好以后会在同级目录下生成vcpkg.exe文件。编译期间，脚本会自动下载vswhere组件。

3. 使用Vcpkg

3.1. 查看Vcpkg支持的开源库列表

执行命令

```
.\vcpkg.exe search
```

3.2. 安装一个开源库

这里的“安装”其实是指下载和编译。

比如我们需要安装常用的jsoncpp库，那么执行命令

```
.\vcpkg.exe install jsoncpp
```

输出：

```
The following packages will be built and installed:
    jsoncpp[core]:x86-windows
Starting package 1/1: jsoncpp:x86-windows
Building package jsoncpp[core]:x86-windows...
-- CURRENT_INSTALLED_DIR=H:/Repos/vcpkg/installed/x86-windows
-- DOWNLOADS=H:/Repos/vcpkg/downloads
-- CURRENT_PACKAGES_DIR=H:/Repos/vcpkg/packages/jsoncpp_x86-windows
-- CURRENT_BUILDTREES_DIR=H:/Repos/vcpkg/buildtrees/jsoncpp
```

```
-- CURRENT_PORT_DIR=H:/Repos/vcpkg/ports/jsoncpp/.
-- Downloading https://github.com/open-source-
parsers/jsoncpp/archive/1.8.1.tar.gz...
-- Downloading https://github.com/open-source-
parsers/jsoncpp/archive/1.8.1.tar.gz... OK
-- Testing integrity of downloaded file...
-- Testing integrity of downloaded file... OK
-- Extracting source H:/Repos/vcpkg/downloads/open-source-parsers-jsoncpp-
1.8.1.tar.gz
-- Extracting done
-- Configuring x86-windows-rel
-- Configuring x86-windows-rel done
-- Configuring x86-windows-dbg
-- Configuring x86-windows-dbg done
-- Build x86-windows-rel
-- Build x86-windows-rel done
-- Build x86-windows-dbg
-- Build x86-windows-dbg done
-- Performing post-build validation
-- Performing post-build validation done
Building package jsoncpp[core]:x86-windows... done
Installing package jsoncpp[core]:x86-windows...
Installing package jsoncpp[core]:x86-windows... done
Elapsed time for package jsoncpp:x86-windows: 47.81 s

Total elapsed time: 47.81 s

The package jsoncpp:x86-windows provides CMake targets:

    find_package(jsoncpp REQUIRED)
    target_link_libraries(main PRIVATE jsoncpp_lib)
```

我们大致可以了解到install会经历这几个过程：

环境初始化

1. 下载源代码（如果已经在cache中，则不下载）
2. 校验文件有效性
3. 解压缩源代码
4. 利用配套工具配置源码工程，在这里是使用的是cmake（如果是ffmpeg，则用msys2）
5. 编译源码。一般会同时编译Release和Debug版本。
6. 把编译好的文件拷贝到相关目录中去（一般是installed目录） 注意点：

如果电脑中没有安装cmake，vcpkg会自动下载portable版本的cmake。但是由于各种原因，下载的网速很慢，所以建议先自行下载安装msi版本的cmake。最好是下载最新版本的cmake。

3.3. 指定编译某种架构的程序库

如果不指定安装的架构，vcpkg默认把开源库编译成x86的Windows版本的库。那vcpkg总共支持多少种架构呢？我们可以使用如下命令便知：

```
.\vcpkg.exe help triplet
```

我们可以看到会列出如下清单：

- arm-uwp
- arm-windows
- arm64-uwp
- arm64-windows
- x64-uwp
- x64-windows-static
- x64-windows
- x86-uwp
- x86-windows-static
- x86-windows

这个清单以后随着版本的迭代还会再增加。vcpkg不仅支持x86架构，还支持arm架构。注意：这里的arm架构特指类似于surface这种运行在arm处理器上的Win10平台，而并非我们传统意义上的Linux或android的ARM平台。

那如果要安装编译某一个架构的开源库，我们该怎么写呢？我们只需要在需要安装的包后面指定相应的triplet即可。例如我们需要编译64位版本的jsoncpp，那么执行如下命令即可。

```
.\vcpkg.exe install jsoncpp:x64-windows
```

3.4. 移除一个已经安装（编译）的开源库

如果移除一个已经安装的开源库，那么执行remove指令即可。比如我们要移除jsoncpp，那么执行命令：

```
.\vcpkg.exe remove jsoncpp
```

注意：

- 这个时候只是移除了默认的x86-winodws版本的文件，如果有其他平台的版本需要移除，需要制定相应的triplet。
- 移除也只是移除了二进制程序库而已，源码包和解压缩的源码并没有删除。

如果想要一键移除“过时”的包，执行命令：

```
.\vcpkg.exe remove --outdated
```

3.5. 列出已经安装的开源库

执行list指令即可，例如：

```
.\vcpkg.exe list
```

3.6. 更新已经安装的开源库

一般有两种更新方式。一个是update指令，可以显示可以升级的开源库的列表。另一个是upgrade的指令，会重新编译所有需要更新的包。

3.7. 导出已经安装的开源库

有的时候，一个项目组中有很多人，不需要每个人都参与编译。一个人编译好所有开源库后到处给别人即可。有的时候也是出于备份的目的，也会导出已经安装的开源库。导出可以执行export指令。例如，我要导出jsoncpp库，那么执行：

```
.\vcpkg.exe export jsoncpp --7zip
```

注意，导出时必须指定导出的包格式。vcpkg支持5种导出包格式，有：

参数	格式
–raw	以不打包的目录格式导出
–nuget	以nuget包形式导出
–ifw	我也不知道这是啥格式
–zip	以zip压缩包形式导出
–7zip	以7z压缩包形式导出

一般地，导出包的格式为：vcpkg-export-<日期>-<时间>

默认情况下只会导出x86-windows的包，如果要导出所有包，那需要制定相应的triplet。比如，如果同时导出x86和x64版本的jsoncpp，那执行命令：

```
.\vcpkg.exe export jsoncpp jsoncpp:x64-windows --7zip
```

这个命令等价于：

```
.\vcpkg.exe export jsoncpp:x86-windows jsoncpp:x64-windows --7zip
```

如果要指定输出目录和特定文件名，需使用”–output=”参数

3.8. 导入备份的开源库

导入比较简单，执行import指令即可。例如：

```
.\vcpkg.exe import xxx.7z
```

4. Vcpkg和Visual Studio的集成

4.1. 什么是集成？

上面我们已经安装了一些第三方库，那如何使用呢？常规情况下，我们需要设置include目录、lib目录等，会有很多工作量。Vcpkg提供了一套机制，可以全自动的适配目录，而开发者不需要关心已安装的库的目录在哪里，也不需要设置。这是Vcpkg的一大优势。

4.2. 集成到全局

“集成到全局”适用于Visual Studio开发环境和msbuild命令行。执行命令：

```
.\vcpkg integrate install
```

当出现“Applied user-wide integration for this vcpkg root.”字样的时候，说明已经集成成功。这时候可以在任意的工程中使用安装好的第三方库。

4.3. 移除全局集成

移除全局集成只要执行下列命令即可：

```
.\vcpkg integrate remove
```

4.4. 集成到工程

上面已经可以集成到全局，为什么还要“集成到工程”呢？因为在大部分情况下，我们不希望集成到全局，毕竟有很多第三方库我们希望自定义处理一下，或者干脆不想集成第三方库。那么集成到工程是最灵活的处理方式。也是工程级项目推荐的处理方式。

“集成到工程”是整个vcpkg中最复杂的一项，它需要利用Visual Studio 中的nuget插件来实现。我们接下来一步一步来说。

4.4.1. 生成配置

执行命令

```
.\vcpkg integrate project
```

这时候会在“\scripts\buildsystems”目录下，生成nuget配置文件.
其中是指vcpkg实际所在目录。

4.4.2. 基本配置

1. 打开Visual Studio，点击菜单“工具->NuGet包管理器->程序包管理器设置”，进入设置界面，点击“程序包源”。
2. 点击“加号”增加一个源。修改源的名字为vcpkg。在“源”的选项中点击右侧的“...”选择vcpkg目录下的“scripts\buildsystems”目录，然后点击右侧的“更新按钮”。
3. 点击“确定”，关闭设置对话框。
4. 到此，全局性的设置已经完成，以后不必再重复设置了。

4.4.3. 工程配置

用Visual Studio 打开一个工程或解决方案。右键点击需要设置的工程，选择“管理NuGet程序包”。在右上角的“程序包源”中选择刚刚设置的“vcpkg”。这样在“浏览”选项卡中就可以看到“vcpkg.H.Repos.vcpkg”。点击最右侧的“安装”。这样就可以集成到某个工程了。

4.5. 集成到CMake

最新的Visual Studio 2015和2017大力支持CMake工程，所以对cmake的支持当然不能少。在cmake中集成只要在cmake文件中加入下面这句话即可。

```
-DCMAKE_TOOLCHAIN_FILE=/scripts/buildsystems/vcpkg.cmake”
```

其中是指vcpkg实际所在目录。

4.6. 集成静态库

Vcpkg默认编译链接的是动态库，如果要链接静态库，目前还没有简便的方法。需要做如下操作

用文本方式打开vcxproj工程文件 在xml的段里面增加如下两句话即可

```
<VcpkgTriplet>x86-windows-static</VcpkgTriplet>
<VcpkgEnabled>true</VcpkgEnabled>
```

在CMake中集成静态库，需要额外指令

```
cmake .. -DCMAKE_TOOLCHAIN_FILE=.../vcpkg.cmake -DVCPKG_TARGET_TRIPLET=x86-
windows-static
```

5. 使用Vcpkg时的注意点

- Vcpkg仅支持Visual Studio 2015 update 3及以上版本（包括Visual Studio 2017），究其原因，很可能和c++11的支持度以及集成原理有关系。
- 目前Vcpkg编译静态库，默认只支持MT模式。

6. vcpkg与Clion

1. Go to File -> Settings -> Build, Execution, Deployment -> CMake
2. Choose one profile, and then find "CMake Options".
3. Insert value to the option: -DCMAKE_TOOLCHAIN_FILE=<path_to_vcpkg>/scripts/buildsystems/vcpkg.cmake
4. Please let me know if there is any better way to do this.

```
-DCMAKE_TOOLCHAIN_FILE=H:/libbuild/vcpkg/scripts/buildsystems/vcpkg.cmake
-DVCPKG_TARGET_TRIPLET=x86-windows-static
```

7. 小结

Vcpkg目前还在不断的完善中，但不可否认，它已经极大的减少了我们在项目启动时，准备第三方库的时间。提高了工作效率。按照时髦的话来说，就是避免了重复造轮子。目前Vcpkg已经集成了上百个常用的开源库，而且数量还在不停增长。毕竟是微软旗下的开源项目，质量还是可以得到保障的，完全可以在工业级项目中得以使用。源代码托管在github上，github社区很活跃，有兴趣的朋友也可以参与进来。