Cross-compiling Swift for Windows with `clang`

This document describes how to cross compile Swift for Windows on a non-Windows host. For more context on the status of Swift on Windows in general, see Getting Started with Swift on Windows

1. Set up Visual Studio environment variables

Building for Windows requires that the Visual Studio environment variables are setup similar to the values on Windows. Currently, the runtime has been tested to build against the Windows 10 SDK at revision 10.10.586.

```
# Visual Studio 2015 does not have VCToolsInstallDir, use VCINSTALLDIR's value
export UCRTVersion=10.0.10586.0
export UniversalCRTSdkDir=".../Windows Kits/10"
export VCToolsInstallDir=".../Microsoft Visual Studio/2017/Community"
```

2. Set up the `visualc` and `ucrt` modules

The `ucrt.modulemap` located at `swift/stdlib/public/Platform/ucrt.modulemap` needs to be copied into `${UniversalCRTSdkDir}/Include/${UCRTVersion}/ucrt` as `module.modulemap`. The `visualc.modulemap` located at `swift/stdlib/public/Platform/visualc.module` needs to be copied into `${VCToolsInstallDir}/include` as `module.modulemap`

3. Configure the runtime to be built with the just built `clang`

Ensure that we use the tools from the just built LLVM and `clang` tools to build the Windows SDK. You will need to pass a few extra options to cmake via the `build-script` invocation to achieve this. You will need to expand out the path where `llvm-ar` and `llvm-ranlib` are built. These are needed to correctly build the static libraries. Note that cross-compiling will require the use of `lld`. Ensure that `lld-link.exe` is available to clang via your path.

macOS

```
--extra-cmake-options=-DSWIFT_BUILD_RUNTIME_WITH_HOST_COMPILER=FALSE,\
-DCMAKE_AR=<path to llvm-ar>,\
-DCMAKE_RANLIB=<path to llvm-ranlib>,\
-DSWIFT_SDKS='OSX;WINDOWS'
```

Linux For Linux, you will need to build the Linux SDK instead of the macOS SDK by replacing the cmake option with `-DSWIFT_SDKS='LINUX;WINDOWS'`.

4. Build the Swift runtime and standard library with `ninja`

From the build directory, you can build the Swift runtime and standard library for Windows using `ninja swiftCore-windows-x86_64`.