Building Swift on Windows

Visual Studio 2017 or newer is needed to build Swift on Windows. The free Community edition is sufficient to build Swift.

The commands below (with the exception of installing Visual Studio) must be entered in the "**x64 Native** Tools Command Prompt for VS2017" (or VS2019, VS2019 Preview depending on the Visual Studio that you are using) in the Start Menu. This sets environment variables to select the correct target platform.

Install dependencies

Visual Studio

An easy way to get most of the tools to build Swift is using the Visual Studio installer. This command installs all needed Visual Studio components as well as Python, Git, CMake and Ninja:

```
curl.exe -sOL https://aka.ms/vs/16/release/vs_community.exe
vs_community ^
  --add Component.CPython3.x64 ^
  --add Microsoft.VisualStudio.Component.Git ^
  --add Microsoft.VisualStudio.Component.VC.ATL ^
  --add Microsoft.VisualStudio.Component.VC.CMake.Project ^
  --add Microsoft.VisualStudio.Component.VC.Tools.x86.x64 ^
  --add Microsoft.VisualStudio.Component.Windows10SDK ^
  --add Microsoft.VisualStudio.Component.Windows10SDK.17763
del /q vs_community.exe
```

If you prefer you can install everything by hand, but make sure to include "Programming Languages|Visual C++" and "Windows and Web Development|Universal Windows App Development|Windows SDK" in your installation. The components listed above are required.

The following link helps in finding the component name given its ID for Visual Studio 2019.

Python

The command above already installs Python 3. Alternatively, in the Visual Studio installation program, under *Individual Components*, install *Python 3 64 bits (3.7.x)*.

If you are building a debug version of Swift, you should also install the Python debug binaries.

1. In the Windows settings, go to *Add and Remove Programs*
2. Select the *Python 3.7.x (64-bit)* entry
3. Click *Modify*, then *Yes*, then *Modify* again and then *Next*
4. Select *Download debug binaries (requires VS 2015 or later)*
5. Click *Install*

Enable Developer Mode

From the settings application, go to `Update & Security`. In the `For developers` tab, select `Developer Mode` for `Use Developer Features`. This is required to enable the creation of symbolic links.

Clone the repositories

1. Clone `swift/main` branch of `apple/llvm-project` into the build workspace
2. Clone `apple/swift-cmark`, `apple/swift`, `apple/swift-corelibs-libdispatch`, `apple/swift-corelibs-foundation`, `apple/swift-corelibs-xctest`, `apple/swift-tools-support-core`, `apple/swift-llbuild`, `apple/swift-argument-parser`, `apple/swift-driver`, `apple/swift-package-manager`, `JPSim/Yams`, `apple/indexstore-db` into the build workspace

- Currently, other repositories in the Swift project have not been tested and may not be supported.

This guide assumes your sources live at the root of `S:`. If your sources live elsewhere, you can create a substitution for this:

```
subst S: <path to sources>
```

```
S:
git clone https://github.com/apple/llvm-project --branch swift/main llvm-project
git clone -c core.autocrlf=input -c core.symlinks=true
https://github.com/apple/swift swift
git clone https://github.com/apple/swift-cmark cmark
git clone https://github.com/apple/swift-corelibs-libdispatch swift-corelibs-
libdispatch
git clone https://github.com/apple/swift-corelibs-foundation swift-corelibs-
foundation
git clone https://github.com/apple/swift-corelibs-xctest swift-corelibs-xctest
git clone https://github.com/apple/swift-tools-support-core swift-tools-support-core
git clone -c core.symlinks=true https://github.com/apple/swift-llbuild swift-llbuild
git clone https://github.com/JPSim/Yams Yams
git clone https://github.com/apple/swift-driver swift-driver
git clone https://github.com/apple/swift-argument-parser swift-argument-parser
git clone -c core.autocrlf=input https://github.com/apple/swift-package-manager
swift-package-manager
git clone https://github.com/apple/indexstore-db indexstore-db
```

Dependencies (ICU, SQLite3, curl, libxml2 and zlib)

The instructions assume that the dependencies are in `S:/Library`. The directory structure should resemble:

```
/Library
  ┣ icu-67
  ┃   ┗ usr/...
  ┣ libcurl-development
  ┃   ┗ usr/...
  ┣ libxml2-development
  ┃   ┗ usr/...
  ┣ sqlite-3.28.0
  ┃   ┗ usr/...
  ┗ zlib-1.2.11
      ┗ usr/...
```

Note that ICU is only required for building Foundation, and SQLite is only needed for building llbuild and onwards. The ICU project provides binaries, alternatively, see the ICU project for details on building ICU from source.

One-time Setup (re-run on Visual Studio upgrades)

Set up the `ucrt`, `visualc`, and `WinSDK` modules by:

- copying `ucrt.modulemap` located at `swift/stdlib/public/Platform/ucrt.modulemap` into `${UniversalCRTSdkDir}/Include/${UCRTVersion}/ucrt` as `module.modulemap`
- copying `visualc.modulemap` located at `swift/stdlib/public/Platform/visualc.modulemap` into `${VCToolsInstallDir}/include` as `module.modulemap`
- copying `winsdk.modulemap` located at `swift/stdlib/public/Platform/winsdk.modulemap` into `${UniversalCRTSdkDir}/Include/${UCRTVersion}/um`
- and setup the `visualc.apinotes` located at `swift/stdlib/public/Platform/visualc.apinotes` into `${VCToolsInstallDir}/include` as `visualc.apinotes`

```
mklink "%UniversalCRTSdkDir%\Include\%UCRTVersion%\ucrt\module.modulemap"
S:\swift\stdlib\public\Platform\ucrt.modulemap
mklink "%UniversalCRTSdkDir%\Include\%UCRTVersion%\um\module.modulemap"
S:\swift\stdlib\public\Platform\winsdk.modulemap
mklink "%VCToolsInstallDir%\include\module.modulemap"
S:\swift\stdlib\public\Platform\visualc.modulemap
mklink "%VCToolsInstallDir%\include\visualc.apinotes"
S:\swift\stdlib\public\Platform\visualc.apinotes
```

Warning: Creating the above links usually requires administrator privileges. The quick and easy way to do this is to open a second developer prompt by right clicking whatever shortcut you used to open the first one, choosing Run As Administrator, and pasting the above commands into the resulting window. You can then close the privileged prompt; this is the only step which requires elevation.

Build the toolchain

```
cmake -B "S:\b\1" ^
  -C S:\swift\cmake\caches\Windows-x86_64.cmake ^
  -D CMAKE_BUILD_TYPE=Release ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_C_COMPILER=cl ^
  -D CMAKE_C_FLAGS="/GS- /Oy /Gw /Gy" ^
  -D CMAKE_CXX_COMPILER=cl ^
  -D CMAKE_CXX_FLAGS="/GS- /Oy /Gw /Gy" ^
  -D CMAKE_EXE_LINKER_FLAGS="/INCREMENTAL:NO" ^
  -D CMAKE_MT=mt ^
  -D CMAKE_SHARED_LINKER_FLAGS="/INCREMENTAL:NO" ^
  -D LLVM_DEFAULT_TARGET_TRIPLE=x86_64-unknown-windows-msvc ^
  -D LLVM_ENABLE_PDB=YES ^
  -D LLVM_EXTERNAL_CMARK_SOURCE_DIR=S:\cmark ^
  -D LLVM_EXTERNAL_SWIFT_SOURCE_DIR=S:\swift ^
  -D SWIFT_PATH_TO_LIBDISPATCH_SOURCE=S:\swift-corelibs-libdispatch ^
  -G Ninja ^
  -S S:\llvm-project\llvm

ninja -C S:\b\1
```

> ***NOTE:*** *Linking with debug information (* `-D LLVM_ENABLE_PDB=YES` *) is very memory intensive. When building with parallel jobs, it is possible to consume upwards of 32 GiB of RAM. You can append* `-D LLVM_PARALLEL_LINK_JOBS=N -D DLLVM_PARALLEL_LINK_JOBS=N` *to reduce the number of parallel link operations to* `N` *which should help reduce the memory pressure. You may need to set this to a low number (e.g. 1) if you see build failures due to memory exhaustion.*

Running Swift tests on Windows

```
path S:\Library\icu-67\usr\bin;S:\b\1\bin;S:\b\1\tools\swift\libdispatch-windows-
x86_64-prefix\bin;%PATH%;%ProgramFiles%\Git\usr\bin
ninja -C S:\b\1 check-swift
```

Build swift-corelibs-libdispatch

```
cmake -B S:\b\2 ^
  -D CMAKE_BUILD_TYPE=RelWithDebInfo ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_C_COMPILER=S:/b/1/bin/clang-cl.exe ^
  -D CMAKE_CXX_COMPILER=S:/b/1/bin/clang-cl.exe ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D ENABLE_SWIFT=YES ^
  -G Ninja ^
  -S S:\swift-corelibs-libdispatch

ninja -C S:\b\2
```

Test swift-corelibs-libdispatch

```
ninja -C S:\b\2 check
```

Build swift-corelibs-foundation

```
cmake -B S:\b\3 ^
  -D CMAKE_BUILD_TYPE=RelWithDebInfo ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_C_COMPILER=S:/b/1/bin/clang-cl.exe ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D CURL_LIBRARY="S:/Library/libcurl-development/usr/lib/libcurl.lib" ^
  -D CURL_INCLUDE_DIR="S:/Library/libcurl-development/usr/include" ^
  -D ICU_I18N_LIBRARY_RELEASE=S:\library\icu-67\usr\lib\icuin67.lib ^
  -D ICU_ROOT=S:\Library\icu-67\usr ^
  -D ICU_UC_LIBRARY_RELEASE=S:\Library\icu-67\usr\lib\icuuc67.lib ^
  -D LIBXML2_DEFINITIONS="/DLIBXML_STATIC" ^
  -D LIBXML2_LIBRARY=S:\Library\libxml2-development\usr\lib\libxml2s.lib ^
  -D LIBXML2_INCLUDE_DIR=S:\Library\libxml2-development\usr\include\libxml2 ^
  -D ENABLE_TESTING=NO ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -G Ninja ^
  -S S:\swift-corelibs-foundation

ninja -C S:\b\3
```

- Add Foundation to your path:

```
path S:\b\3\bin;%PATH%
```

Build swift-corelibs-xctest

```
cmake -B S:\b\4 ^
  -D CMAKE_BUILD_TYPE=RelWithDebInfo ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -D Foundation_DIR=S:\b\3\cmake\modules ^
  -D LIT_COMMAND=S:\llvm-project\llvm\utils\lit\lit.py ^
  -G Ninja ^
  -S S:\swift-corelibs-xctest

ninja -C S:\b\4
```

- Add XCTest to your path:

```
path S:\b\4;%PATH%
```

Test XCTest

```
ninja -C S:\b\4 check-xctest
```

Rebuild Foundation

```
cmake -B S:\b\3 ^
  -D CMAKE_BUILD_TYPE=RelWithDebInfo ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_C_COMPILER=S:/b/1/bin/clang-cl.exe ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D CURL_LIBRARY="S:/Library/libcurl-development/usr/lib/libcurl.lib" ^
  -D CURL_INCLUDE_DIR="S:/Library/libcurl-development/usr/include" ^
  -D ICU_I18N_LIBRARY_RELEASE=S:\library\icu-67\usr\lib\icuin67.lib ^
  -D ICU_ROOT=S:\Library\icu-67\usr ^
  -D ICU_UC_LIBRARY_RELEASE=S:\Library\icu-67\usr\lib\icuuc67.lib ^
  -D LIBXML2_LIBRARY=S:\Library\libxml2-development\usr\lib\libxml2s.lib ^
  -D LIBXML2_INCLUDE_DIR=S:\Library\libxml2-development\usr\include\libxml2 ^
  -D LIBXML2_DEFINITIONS="/DLIBXML_STATIC" ^
  -D ENABLE_TESTING=YES ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -D XCTest_DIR=S:\b\4\cmake\modules ^
  -G Ninja ^
  -S S:\swift-corelibs-foundation

ninja -C S:\b\3
```

Test Foundation

```
ninja -C S:\b\3 test
```

Build swift-tools-core-support

```
cmake -B S:\b\5 ^
  -D BUILD_SHARED_LIBS=YES ^
  -D CMAKE_BUILD_TYPE=RelWithDebInfo ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_C_COMPILER=S:/b/1/bin/clang-cl.exe ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -D Foundation_DIR=S:\b\3\cmake\modules ^
  -D SQLite3_INCLUDE_DIR=S:\Library\sqlite-3.28.0\usr\include ^
  -D SQLite3_LIBRARY=S:\Library\sqlite-3.28.0\usr\lib\SQLite3.lib ^
  -G Ninja ^
  -S S:\swift-tools-support-core

ninja -C S:\b\5
```

Build swift-llbuild

```
cmake -B S:\b\6 ^
  -D BUILD_SHARED_LIBS=YES ^
  -D CMAKE_BUILD_TYPE=RelWithDebInfo ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_CXX_COMPILER=S:/b/1/bin/clang-cl.exe ^
  -D CMAKE_CXX_FLAGS="-Xclang -fno-split-cold-code" ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D LLBUILD_SUPPORT_BINDINGS=Swift ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -D Foundation_DIR=S:\b\3\cmake\modules ^
  -D SQLite3_INCLUDE_DIR=S:\Library\sqlite-3.28.0\usr\include ^
  -D SQLite3_LIBRARY=S:\Library\sqlite-3.28.0\usr\lib\sqlite3.lib ^
  -G Ninja ^
  -S S:\swift-llbuild

ninja -C S:\b\6
```

- Add llbuild to your path:

```
path S:\b\6\bin;%PATH%
```

Build Yams

```
cmake -B S:\b\7 ^
  -D BUILD_SHARED_LIBS=YES ^
  -D CMAKE_BUILD_TYPE=Release ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -D Foundation_DIR=S:\b\3\cmake\modules ^
  -D XCTest_DIR=S:\b\4\cmake\modules ^
  -G Ninja ^
  -S S:\Yams

ninja -C S:\b\7
```

Build swift-argument-parser

```
cmake -B S:\b\8 ^
  -D BUILD_SHARED_LIBS=YES ^
  -D CMAKE_BUILD_TYPE=Release ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -D Foundation_DIR=S:\b\3\cmake\modules ^
  -D XCTest_DIR=S:\b\4\cmake\modules ^
  -G Ninja ^
  -S S:\swift-argument-parser

ninja -C S:\b\8
```

Build swift-driver

```
cmake -B S:\b\9 ^
  -D BUILD_SHARED_LIBS=YES ^
  -D CMAKE_BUILD_TYPE=Release ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -D Foundation_DIR=S:\b\3\cmake\modules ^
  -D TSC_DIR=S:\b\5\cmake\modules ^
  -D LLBuild_DIR=S:\b\6\cmake\modules ^
  -D Yams_DIR=S:\b\7\cmake\modules ^
  -D ArgumentParser_DIR=S:\b\8\cmake\modules ^
  -G Ninja ^
  -S S:\swift-driver

ninja -C S:\b\9
```

Build swift-package-manager

```
cmake -B S:\b\10 ^
  -D BUILD_SHARED_LIBS=YES ^
  -D CMAKE_BUILD_TYPE=Release ^
  -D CMAKE_C_COMPILER=S:/b/1/bin/clang-cl.exe ^
  -D CMAKE_INSTALL_PREFIX=C:\Library\Developer\Toolchains\unknown-Asserts-
development.xctoolchain\usr ^
  -D CMAKE_MT=mt ^
  -D CMAKE_Swift_COMPILER=S:/b/1/bin/swiftc.exe ^
  -D dispatch_DIR=S:\b\2\cmake\modules ^
  -D Foundation_DIR=S:\b\3\cmake\modules ^
  -D TSC_DIR=S:\b\5\cmake\modules ^
  -D LLBuild_DIR=S:\b\6\cmake\modules ^
  -D Yams_DIR=S:\b\7\cmake\modules ^
  -D ArgumentParser_DIR=S:\b\8\cmake\modules ^
  -D SwiftDriver_DIR=S:\b\9\cmake\modules ^
  -G Ninja ^
  -S S:\swift-package-manager

ninja -C S:\b\10
```

Indicate to swift-package-manager where to find the Package Description before installation:

```
set SWIFTPM_PD_LIBS=S:\b\10\pm
```

Install the Swift toolchain on Windows

- Run ninja install:

```
ninja -C S:\b\1 install
```

- Add the Swift on Windows binaries path (`C:\Library\Developer\Toolchains\unknown-Asserts-development.xctoolchain\usr\bin`) to the `PATH` environment variable.