Getting started with Swift on OpenBSD

Swift builds and runs on OpenBSD (tested on 6.8), with some special considerations.

Preparing

The following packages are required to build Swift. You can install these via `pkg_add`:

```
$ doas pkg_add bash cmake e2fsprogs git icu4c ninja py-six python3
```

Because LLVM is built as part of building Swift and does not include some of the patches to handle the OpenBSD library naming convention, you will need to create some symlinks:

```
$ doas ln -s /usr/lib/libc++abi.so.2.1 /usr/lib/libc++abi.so
$ doas ln -s /usr/lib/libc++.so.4.0 /usr/lib/libc++.so
$ doas ln -s /usr/lib/libc.so.96.0 /usr/lib/libc.so
$ doas ln -s /usr/lib/libm.so.10.1 /usr/lib/libm.so
$ doas ln -s /usr/local/lib/libicuuc.so.18.0 /usr/local/lib/libicuuc.so
$ doas ln -s /usr/lib/libpthread.so.26.1 /usr/lib/libpthread.so
```

Note: you may need to update the version numbers if necessary.

Also link `~/bin/python` to the `python2.7` binary:

```
$ doas ln -s /usr/local/bin/python2.7 ~/bin/python
```

Since the build requires significant amounts of memory at certain points, you may need to ensure that the user you are using to build Swift has the appropriate limits set. Using the `staff` group in `login.conf` and ensuring the shell limits are raised is recommended.

Downloading the source

Download the sources with the Getting Started guide (see "Cloning the project"). Use the config file below when running `update-checkout` by specifying the file name with the `--config` flag. This config file just prepares cmark, LLVM, and Swift (as this is the minimal set of dependencies which has been tested for OpenBSD).

```
{
  "ssh-clone-pattern": "git@github.com:%s.git",
  "https-clone-pattern": "https://github.com/%s.git",
  "default-branch-scheme": "main",
  "repos": {
    "cmark": { "remote": { "id": "apple/swift-cmark" } },
    "llvm-project": { "remote": { "id": "apple/llvm-project" } },
    "swift": { "remote": { "id": "apple/swift" } }
  },
  "branch-schemes": {
    "main": {
      "aliases": [ "main", "swift/main" ],
      "repos": {
        "cmark": "main",
        "llvm-project": "swift/main",
        "swift": "main"
      }
    }
  }
}
```

Building

Once the sources have completed downloading, you can use the standard `build-script` mechanism to start building Swift. However, some options are required to be set to successfully build Swift.

These options are:

- `--skip-build-clang-tools-extra` and `--skip-build-compiler-rt` : to ensure LLVM builds cleanly,
- `--extra-cmake-options=`
  - `-DCMAKE_DISABLE_FIND_PACKAGE_Backtrace=TRUE,-DCMAKE_DISABLE_FIND_PACKAGE_LibXml2=TRUE,-DLLVM_VERSION_SUFFIX=''` : to ensure LLVM builds cleanly,
  - `-DSWIFT_BUILD_SOURCEKIT=OFF,-DSWIFT_BUILD_SYNTAXPARSERLIB=OFF,-DSWIFT_ENABLE_EXPERIMENTAL_CONCURRENCY=OFF` : to ensure Swift does not attempt to build libdispatch, which is not yet supported on OpenBSD,
  - `-DSWIFT_USE_LINKER=lld` : to specify that `lld` should be used over `gold` ,
  - `-DCMAKE_INSTALL_DIR=/usr/local"` : to set the correct platform install directory.

In full, the minimal set of flags to supply to `build-script` looks like:

```
$ ./utils/build-script \
    --skip-build-clang-tools-extra \
    --skip-build-compiler-rt \
    --extra-cmake-options="\
        -DCMAKE_DISABLE_FIND_PACKAGE_Backtrace=TRUE,\
        -DCMAKE_DISABLE_FIND_PACKAGE_LibXml2=TRUE,\
        -DLLVM_VERSION_SUFFIX='',\
        -DSWIFT_BUILD_SOURCEKIT=OFF,\
        -DSWIFT_BUILD_SYNTAXPARSERLIB=OFF,\
        -DSWIFT_ENABLE_EXPERIMENTAL_CONCURRENCY=OFF,\
        -DSWIFT_IMPLICIT_CONCURRENCY_IMPORT=OFF,\
        -DSWIFT_USE_LINKER=lld,\
        -DCMAKE_INSTALL_DIR=/usr/local"
```

You may wish to also supply the flag `--llvm-targets-to-build=host` , to speed up the LLVM build slightly.

For debug builds especially, consider also installing the `llvm` package and setting `-DCMAKE_AR=/usr/local/bin/llvm-ar` and `-DCMAKE_RANLIB=/usr/local/bin/llvm-ranlib` with the `extra-cmake-options` flag, to work around problems creating indexes to archives containing object files with large numbers of section headers.