Unified Headers

Issue #120

Before NDK r14, we had a set of libc headers for each API version. In many cases these headers were incorrect. Many exposed APIs that didn't exist, and others didn't expose APIs that did.

In NDK r14 (as an opt in feature) we unified these into a single set of headers. In r15 these are used by default. This single header path is used for *every* platform level. API level guards are handled with `#ifdef`. These headers can be found in prebuilts/ndk/headers.

Unified headers are built directly from the Android platform, so they be up to date and correct (or at the very least, any bugs in the NDK headers will also be a bug in the platform headers, which means we're much more likely to find them).

Known Issues

- Some third-party projects have incorrect feature checks for things like `epoll_create1`. These are not bugs in the NDK, but rather need to be fixed in those projects. Boost and libev are projects that we know are affected by this. We'll be sending some patches to those projects soon. See bug 302 and bug 394.
- Standalone toolchains using GCC are not supported out of the box. To use GCC, pass `-D__ANDROID_API__=$API` when compiling. Note: this is not something we will be fixing.

Using Unified Headers

Unified headers are enabled by default starting with NDK r15. If your build is not yet compatible with unified headers, you can revert to the deprecated headers. The way to do so depends on your build system.

Warning: The deprecated headers will be removed from the NDK in r16. If you need to revert to the deprecated headers, make sure you're working on fixing your build or filing bugs.

ndk-build

Add the following to your Application.mk:

```
APP_DEPRECATED_HEADERS := true
```

CMake

```
cmake -DANDROID_DEPRECATED_HEADERS=ON ...
```

Standalone Toolchains

```
$NDK/build/tools/make_standalone_toolchain.py --deprecated-headers ...
```

For general standalone toolchain documentation, see https://developer.android.com/ndk/guides/standalone_toolchain.html

Gradle Experimental Plugin

```
model {
    android {
        ndk {
            useUnifiedHeaders false
        }
    }
}
```

Using unified headers with `configure && make`

If you're trying to build a traditional open-source `configure && make` style project with the NDK, unified headers are your best choice. The basic steps look like this (here for 32-bit ARM):

```
$ $NDK/build/tools/make_standalone_toolchain.py --unified-headers \
    --arch arm --api 21 --install-dir /tmp/ndk-arm-21
$ cd <project>
$ export CC=/tmp/ndk-arm-21/arm-linux-androideabi-clang
$ export LDFLAGS="-pie"
$ ./configure --host=arm-linux-androideabi
$ make
```

Replace “arm” and “arm-linux-androideabi” with the appropriate pair for the architecture you actually want to build for, and 21 with the API level you want to target.

Supporting Unified Headers in Your Build System

App developers can stop reading here. The following information is only relevant to build system engineers.

Unified headers require only a few changes compared to using the deprecated NDK headers. For reference, this patch added support to ndk-build: https://android-review.googlesource.com/c/239934/

1. The compile time sysroot is now `$NDK/sysroot`. Previously this was `$NDK/platforms/android-$API/arch-$ARCH`.
2. Pass `-isystem $NDK/sysroot/usr/include/$TRIPLE` when compiling. The triple has the following mapping:

Arch	Triple

Arch	Triple
ARM	`arm-linux-androideabi`
ARM64	`aarch64-linux-android`
MIPS	`mipsel-linux-android`
MIPS64	`mips64el-linux-android`
x86	`i686-linux-android`
x86_64	`x86_64-linux-android`

This is needed for architecture specific headers such as those in `asm/` and `machine/`. We plan to teach Clang's driver to automatically search the architecture specific include directory, but that has yet to be done.

3. Pass `-D__ANDROID_API__=$API` when compiling. This define used to be provided by `<android/api-level.h>`, but with only one set of headers this is no longer possible. In the future we will look in to adding `-mandroid-version` or similar to Clang so this is automatic.
4. At link time, change nothing. All link time build behavior should match the deprecated headers behavior. `--sysroot` should still point to `$NDK/platforms/android-$API/arch-$ARCH/`.