AOSP's switch to Clang

Bernhard "Bero" Rosenkränzer <bero@linaro.org>

LMG
Mobile

2016-11-03

Quick overview

- With the 7.0 (N) release, AOSP uses clang instead of gcc to compile its userland.
- Required mostly small changes:
  - Enforcing code follows the C and C++ standards rather than "gcc takes it, so it's ok"
  - Pain points: Get rid of VLAIS (Variable-Length Arrays in Structs), c89 vs. c99 definition of "extern", inline assembly not using pre-unified syntax
  - Several "real" bugs detected because of additional warnings and errors from clang

    -
      ```
      void doSomething(char a[10]) {
              if(memcmp(a, something, sizeof(a))) { … }
      }
      ```
- Performance of OS similar to before the switch, build time faster
- gcc still used to build the kernel and some HALs for old devices

The kernel

- Work on getting the kernel to build (and work) with clang is going on:
- git://android-git.linaro.org/kernel/hikey-clang.git
  android-hikey-linaro-4.4-clang branch
- Slightly outdated, will be rebased to 4.9 soon
- Some patches are ready to go upstream, some others are horrible hacks...

The kernel

- Need to use -no-integrated-as for now - numerous gas-isms
- With released versions of clang, needs clang patch for profiler function naming on arm and aarch64 (`mcount` vs. `__gnu_mcount_nc` vs. `_mcount`). This is fixed in clang 4.0 snapshots (needs new parameter: `-meabi gnu`)
- ld from binutils 2.27 is too strict on .cfi_sections -- use 2.26 or a 2.28 snapshot (either will work)

The kernel

- ARMv8 Hardware crypto disabled for now (doesn’t compile):

```
#define ASM_EXPORT(sym, val) \
    asm(".globl " #sym "; .set " #sym ", %0" :: "I"(val));
ASM_EXPORT(sha256_ce_offsetof_count,
    offsetof(struct sha256_ce_state, sst.count));
```

- Generated code (clang):

```
.globl sha256_ce_offsetof_count; .set sha256_ce_offsetof_count, #32
```

- Generated code (gcc):

```
.globl sha256_ce_offsetof_count; .set sha256_ce_offsetof_count, 32
```

The kernel

- Docker container with pre-patched clang and source tree is available:
  - `docker run -ti bero/hikey-kernel-clang`
  - will be updated to clang 4.0 snapshot soon (currently on 3.8.1)
- System runs and works, but some unexpected side effects (firefox crashes on startup) that need to be investigated
- No official plans (that I know of) to build AOSP kernels with clang, but hikey might lead the way at some point
- Should make life easier for 32-bit compat vDSO in aarch64 kernel - clang host compiler is the same as cross compiler

Staying on top of clang

- All clang patches needed to build AOSP have been upstreamed
- Linaro is running nightly builds of AOSP master with clang master in CI:
  - https://ci.linaro.org/job/android-master-clang/
  - Produces working builds, we get notified when things break
  - Pain point: AOSP master still frequently not in sync with Android master. Things tend to break after new upstream releases, patches we submit to AOSP master to fix things not always useful.
- Will start investigating new compiler flags, and plugins like polly

New warnings (and errors) with clang 4.0

- -Wexpansion-to-defined [in libchrome, gtest]

```
#define USE_X (defined(A) || defined(B))
```

⇩

```
#if (defined(A) || defined(B))
#define USE_X 1
#else
#define USE_X 0
#endif
```

- external/vixl/src/vixl/a64/disasm-a64.h:150:48: error: 'format' attribute argument not supported: gnu_printf [-Werror,-Wignored-attributes]

New warnings (and errors) with clang 4.0

- Taking address of packed member may result in an unaligned pointer value (-Waddress-of-packed-member) [numerous places in art]
- art/runtime/mirror/array.h:160:11: error: instantiation of variable 'art::mirror::PrimitiveArray<unsigned short>::array_class_' required here, but no definition is available [-Werror,-Wundefined-var-template]
- art/runtime/base/arena_allocator.cc:146:16: error: explicit instantiation of 'ArenaAllocatorStatsImpl<false>' that occurs after an explicit specialization has no effect [-Werror,-Winstantiation-after-specialization]
- art/cmdline/cmdline_parser.h:273:16: error: binding dereferenced null pointer to reference has undefined behavior [-Werror,-Wnull-dereference]

  ```
  return *reinterpret_cast<TArg*>(0);  // Blow up.
  ```

  -Werror=null-dereference in GLOBAL_CLANG_CFLAGS_NO_OVERRIDE

New warnings (and errors) with clang 4.0

- art/compiler/utils/arm/assembler_thumb2.cc:327:26: error: implicit conversion from 'int' to 'int16_t' (aka 'short') changes value from 49152 to -16384 [-Werror,-Wconstant-conversion]:
  ```
  int16_t encoding = B15 | B14;
  ```
- -Wno-varargs (system/netd), should be fixed properly
- __gcc_atomic vs. std::__gcc_atomic in libc++ -- fix is simple, better fix is to keep llvm and libc++ in sync, update prebuilts/clang and external/libcxx at the same time (and libcxxabi, libunwind_llvm)

Staying on top of clang

- Remarkably few bugs for a more or less random snapshot -- but not 0
  https://llvm.org/bugs/show_bug.cgi?id=30900
  https://llvm.org/bugs/show_bug.cgi?id=30908

Linkers

- Current situation: binutils provides BFD linker and Gold, mclinker still in tree, lld starting to show up (empty external/lld directory in master and 7.1.0_r4)
- AOSP binutils tends to be behind upstream releases, not benefitting from e.g. gold-on-aarch64 fixes in 2.27
- Need binutils anyway (for ar, strip, nm, objcopy)

Linkers

- lld probably best way ahead
- Status of lld 4.0 snapshots:
    - Support all architectures relevant to AOSP (Aarch64, ARM, x86, MIPS)
    - Support gc-sections and ICF (Identical Code Folding)
    - Missing support for linker scripts (not used extensively in AOSP, might be sufficient)
    - Different algorithm for detecting symbols in static libraries
    - clang LTO (but not yet gcc LTO) supported
    - Missing:
        - `--fix-cortex-a53-843419` *(actually forced into the linker command line by clang)*
        - `--icf=safe` -- *will --icf=all break anything?*
        - `--exclude-libs=libgcc.a`

Future of gcc in AOSP

- Official AOSP gcc toolchain stuck at 4.9, chances of that changing low to zero
- Linaro still running test builds of AOSP with USE_CLANG_PLATFORM_BUILD=false, using gcc 6.x releases from the Linaro toolchain working group (currently 6.2-16.11-rc1).
- Useful for benchmarking and error detection -- different compilers produce different warnings
- Will likely stop running builds if USE_CLANG_PLATFORM_BUILD parameter disappears upstream and isn't easy to patch back in -- at some point, AOSP codebase may start relying on clang plugins

Linaro

LMG
Mobile

Questions?
Other things we should do to help the effort?

Bernhard “Bero” Rosenkränzer <bero@linaro.org>

2016-11-03