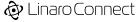


LLVM glibc project

The road to build glibc with clang/llvm Adhemerval Zanella Carlos Seo

Project Goals

- Allow glibc to be built with a different compiler than gcc
 Initially only aarch64 and x86_64
- Improve glibc code coverage by using a different set of compiler warnings
- Improve clang support with features that are tied to compiler support
 i.e. fortify headers
- Check possible performance differences
 - i.e. generic math code



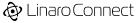
Challenges

- Glibc uses a <u>GNU C standard</u>, so it relies a lot on gcc extensions to the language
- Similar for assembler and static linker, where it assumes binutils
- Some GNU extensions were already discussed and LLVM community would not implement it
 - i.e. nested functions
- Others are not fully compatible between compilers
 - i.e _Float128 support on some ABIs



Initial Work

- Remove the usage of GNU extension that LLVM will not implement and that do not affect code ABI or code generation
 - Nested functions
 - ASM label after first use
 - Instructions not supported by LLVM's integrated assembler
- Adapt glibc to use a different static linker than binutils
 - Remove the use of the linker script not fully supported by all IId versions
 - Do not assume all binutils optimizations
 - i.e. GOT avoid relocation on x86
 - Assume disjointed .rela.dyn and .rela.plt for loader
 - Missing ABI support
 - i.e. ARM TLS descriptors for some old IId versions



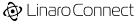
Initial Work

- Refactor code to avoid binutils and/or gcc-specific extensions
 - Use of gcc -fno-toplevel-reorder on errlist-compat.c
 - Remove the usage of a linker script to generate the required RELRO
 - Old IId versions did not support all the required directives
 - Fixes some assembly usages that are not fully compatible with clang integrated assembler
 - i.e. binutils .tfloat directive
- Improve generic clang support
 - Reorganize headers to avoid function redefinition after initial usage
 - Improve fortify support



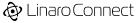
Current status

- Most of the work in out-of-branch (azanella/clang on sourceware.org), which is based on current master branch
 - \circ 1/3 patches to build all glibc modules for aarch64 and x86_64
 - ²/₃ patches to build all required testcases
- All new configure options supported
 - --enable-stack-protector, --enable-fortify-source, --enable-cet,
 --enable-memory-tagging
- Targeting a minimal of clang/llvm version **16**



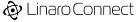
Current status

- It uses the gcc runtime as default (libgcc.a and libgcc_s.so)
 - Using the llvm provided is possible, but requires more work
- clang/llvm version 18 and using gcc 11 runtime, along with llvm tools (lld, llvm-ar, etc)
 - 5 regressions on both x86_64 and aarch64
 - It uses the gcc runtime
 - Most failures are math corner cases
- The bootstrap is still not as easy as gcc



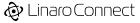
Limitations

- Some iFUNC corner cases do not work correctly when built with clang and linked with binutils (<u>GNU-1045</u>)
 - iFUNC is not widespread, and using clang with IId does not trigger the regressions
- gmon on aarch64 have a different ABI for gcc and clang (<u>GNU-1049</u>)
 - Not sure how often it is used now with more complete profiling solutions like Linux perf and binutils gprofng



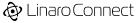
Limitations

- Some math tests show some wrong code generation (GNU-1052)
 - Although limited to long double / Float128
- To use the llvm provided runtime would require more work (GNU-1126).
 - The **Ilvm-libgcc** runtime helps, but it also shows a lot of regressions



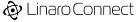
Add support to use compiler-rt/libunwind

- Glibc requires libgcc for floating point operations for long double / _Float128, and unwind, backtrace, and pthread cancellation
 - glibc also expects libgcc.so to provide __gcc_personality_v0, which is used by the GNU cleanup handler extensions (used on pthread cancellation)
 - IIvm libunwind does not provide the __gcc_personality_v0 symbol



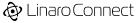
Add support to use compiler-rt/libunwind

- LLVM provides a specific runtime that mimic the gcc one: **Ilvm-libgcc**
 - Enabling it is exclusive with compiler-rt and libunwind
 - It does not provide __sfp_handle_exceptions on x86_64 (which raises floating point exceptions)
 - It does not support floating point rounding mode different than default (FE_TONEAREST), nor floating point exceptions for some modes
 - All symbols are public (different than hidden with libgcc), which generates PLT calls for soft-fp calls
- Issue <u>GNU-1126</u> shows the current status with **Ilvm-libgcc**
 - **96** regressions for x86_64
 - o **355** for aarch64



Future work

- Upstream the first part to enable llvm build aarch64 and x86_64 • It should also enable the build for 32 bit arm (not thumb) and i686
- Upstream the remaining patches to fix the testcases
- Work to fix the regressions with llvm-libgcc
 - Assuming this would be desirable way to bootstrap a clang toolchain, as ChromeOS is aiming
- Add Continuous Integration to build / testing to avoid regressions on both clang and glibc changes





Thank you