Switching to LLVM based toolchains in Yocto based distributions

Bernhard “bero” Rosenkränzer <bero@lindev.ch>

LLVM Distributors Conference 2021

2021-09-16

All Scenarios OS - a new portable OS

All Scenarios OS [working title] is - or will be, a bit further down the road, a new portable OS for anything from small IoT sensors to rich multimedia companions.
As such, the OS environment can sit on multiple different kernels - from Linux for higher end devices to Zephyr, FreeRTOS or Huawei's own LiteOS for the smallest IoT devices.

With reuse of open source components and adherence to open standards as key features and modularity in mind, we decided not to reinvent the wheel, but start with something already good at building minimalistic OSes - Yocto.

To make sure we can work with the community on delivering security updates for a long time, we went with the Long Term Support branch - Dunfell.

But of course that means gcc 9.3... And we'd rather lead than become a museum of outdated components. And we need RISC-V support. And Cortex-M optimizations. And ...

Switching to LLVM toolchain components

2 viable options to get the toolchain to where we need it to be: Update gcc to a more current release, or switch to clang.

The primary toolchain for the first release of All Scenarios OS will be built around LLVM 12.0.1 (an update to 13.0 is possible but not very likely at this point).

Since in the IoT world (unlike on desktops), fortunately we don't have to care about binary compatibility with older distributions a lot, we can go all the way: compiler-rt over libgcc, libc++ over libstdc++ [on devices using C++ at all], and default libc musl 1.2.2 (no need to build glibc, which still requires gcc).

How to do it

There is a clang layer for Yocto - it isn't an official component, but it works well and is actively maintained.
https://github.com/kraj/meta-clang
Simply pull it in and set `TOOLCHAIN="clang"` in `local.conf`, and you're ready to get started.

Not so much fun on Dunfell (LTS) though - the dunfell branch is built around LLVM/Clang 10.
So we took the master branch and brought back Dunfell support. Our working tree lives at https://git.ostc-eu.org/OSTC/OHOS/meta-clang/-/tree/OSTC/clang-12 and has been upstreamed into the "clang12" branch of meta-clang.

OSTC Open Source Technology Center

How to do it

There is a clang layer for Yocto - it isn't an official component, but it works well and is actively maintained.
https://github.com/kraj/meta-clang
Simply pull it in and set `TOOLCHAIN="clang"` in local.conf, and you're ready to get started.

Not so much fun on Dunfell (LTS) though - the dunfell branch is built around LLVM/Clang 10.
So we took the master branch and brought back Dunfell support. Our working tree lives at https://git.ostc-eu.org/OSTC/OHOS/meta-clang/-/tree/OSTC/clang-12 and has been upstreamed into the "clang12" branch of meta-clang.

OSTC Open Source Technology Center

Dealing with glibc and friends

meta-clang introduces a config file in which you can list packages that need to be built with gcc even if the `TOOLCHAIN` variable is set to `clang`.

Outside of being a useful workaround, it is also an interesting place to look for clang developers who want to reduce the number of packages with mandated gcc: https://github.com/kraj/meta-clang/blob/master/conf/nonclangable.conf

Workarounds tend to stay longer than needed

The problem with a workaround that works well: It tends to be forgotten about and stay much longer than necessary...

That certainly goes for `TOOLCHAIN_whatever = "gcc"`

Once we had meta-clang working with clang 12 on top of Dunfell, we looked at those workarounds.

- 7 instances of `TOOLCHAIN_* = "gcc"` removed without needing other changes
- 5 instances of adding `-fno-integrated-as` removed without needing other changes
- 1 instance of making `-fno-integrated-as` a per-architecture flag (GNU as extensions used only on ARM32) instead of using it unconditionally

OSTC Open Source Technology Center

Workarounds tend to stay longer than needed

- Various packages fixed with minor patches — among those an interesting one that has caused unnoticed runtime errors with gcc forever (clang spotted the problem at build time)

```
-{ "unsigned", "variant", "void" "_Bool", "_Complex", "_Imaginary" };
+{ "unsigned", "variant", "void", "_Bool", "_Complex", "_Imaginary" };
```

A couple of takeaways

- Look at what your compiler is telling you instead of complaining about that buggy compiler and running back to what you were using before. It might just be right. (This applies to both different compilers and new compiler versions...)
- Revisit workarounds from time to time. They don't have to stay forever.
- `nonclangable.conf` can be an interesting thing to look at for LLVM developers even if you don't use All Scenarios OS or Yocto.

Thank You!
... And get in touch with us

VISIT OUR GITLAB REPO:
https://git.ostc-eu.org/OSTC

TALK TO US ON MATTERMOST
(requires gitlab account from link above):
https://chat.ostc-eu.org/

AND VISIT OUR WEBSITE:
www.ostc-eu.org

OR WRITE A COMMENT ON THE CFP OR EMAIL
bero@lindev.ch

OSTC
Open Source Technology Center