Building Linux with Clang/LLVM

This document covers how to build the Linux kernel with Clang and LLVM utilities.

About

The Linux kernel has always traditionally been compiled with GNU toolchains such as GCC and binutils. Ongoing work has allowed for Clang and LLVM utilities to be used as viable substitutes. Distributions such as Android, ChromeOS, and OpenMandriva use Clang built kernels. LLVM is a collection of toolchain components implemented in terms of C++ objects. Clang is a front-end to LLVM that supports C and the GNU C extensions required by the kernel, and is pronounced "klang," not "see-lang."

Clang

The compiler used can be swapped out via `CC=` command line argument to `make`. `CC=` should be set when selecting a config and during a build.

```
make CC=clang defconfig

make CC=clang
```

Cross Compiling

A single Clang compiler binary will typically contain all supported backends, which can help simplify cross compiling.

```
make ARCH=arm64 CC=clang CROSS_COMPILE=aarch64-linux-gnu-
```

`CROSS_COMPILE` is not used to prefix the Clang compiler binary, instead `CROSS_COMPILE` is used to set a command line flag: `--target=<triple>`. For example:

```
clang --target=aarch64-linux-gnu foo.c
```

LLVM Utilities

LLVM has substitutes for GNU binutils utilities. They can be enabled individually. The full list of supported make variables:

```
make CC=clang LD=ld.lld AR=llvm-ar NM=llvm-nm STRIP=llvm-strip \
  OBJCOPY=llvm-objcopy OBJDUMP=llvm-objdump READELF=llvm-readelf \
  HOSTCC=clang HOSTCXX=clang++ HOSTAR=llvm-ar HOSTLD=ld.lld
```

To simplify the above command, Kbuild supports the `LLVM` variable:

```
make LLVM=1
```

If your LLVM tools are not available in your PATH, you can supply their location using the LLVM variable with a trailing slash:

```
make LLVM=/path/to/llvm/
```

which will use `/path/to/llvm/clang`, `/path/to/llvm/ld.lld`, etc.

If your LLVM tools have a version suffix and you want to test with that explicit version rather than the unsuffixed executables like `LLVM=1`, you can pass the suffix using the `LLVM` variable:

```
make LLVM=-14
```

which will use `clang-14`, `ld.lld-14`, etc.

`LLVM=0` is not the same as omitting `LLVM` altogether, it will behave like `LLVM=1`. If you only wish to use certain LLVM utilities, use their respective make variables.

The integrated assembler is enabled by default. You can pass `LLVM_IAS=0` to disable it.

Omitting CROSS_COMPILE

As explained above, `CROSS_COMPILE` is used to set `--target=<triple>`.

If `CROSS_COMPILE` is not specified, the `--target=<triple>` is inferred from `ARCH`.

That means if you use only LLVM tools, `CROSS_COMPILE` becomes unnecessary.

For example, to cross-compile the arm64 kernel:

```
make ARCH=arm64 LLVM=1
```

If `LLVM_IAS=0` is specified, `CROSS_COMPILE` is also used to derive `--prefix=<path>` to search for the GNU assembler and linker.

```
make ARCH=arm64 LLVM=1 LLVM_IAS=0 CROSS_COMPILE=aarch64-linux-gnu-
```

Supported Architectures

LLVM does not target all of the architectures that Linux supports and just because a target is supported in LLVM does not mean that the kernel will build or work without any issues. Below is a general summary of architectures that currently work with `CC=clang` or `LLVM=1`. Level of support corresponds to "S" values in the MAINTAINERS files. If an architecture is not present, it either means that LLVM does not target it or there are known issues. Using the latest stable version of LLVM or even the development tree will generally yield the best results. An architecture's `defconfig` is generally expected to work well, certain configurations may have problems that have not been uncovered yet. Bug reports are always welcome at the issue tracker below!

Architecture	Level of support	`make` command
arm	Supported	`LLVM=1`
arm64	Supported	`LLVM=1`
mips	Maintained	`CC=clang`
powerpc	Maintained	`CC=clang`
riscv	Maintained	`CC=clang`
s390	Maintained	`CC=clang`
x86	Supported	`LLVM=1`

Getting Help

- Website
- Mailing List: <llvm@lists.linux.dev>
- Old Mailing List Archives
- Issue Tracker
- IRC: #clangbuiltlinux on irc.libera.chat
- Telegram: @ClangBuiltLinux
- Wiki
- Beginner Bugs

Getting LLVM

- https://releases.llvm.org/download.html
- https://github.com/llvm/llvm-project
- https://llvm.org/docs/GettingStarted.html
- https://llvm.org/docs/CMake.html
- https://apt.llvm.org/
- https://www.archlinux.org/packages/extra/x86_64/llvm/
- https://github.com/ClangBuiltLinux/tc-build
- https://github.com/ClangBuiltLinux/linux/wiki/Building-Clang-from-source
- https://android.googlesource.com/platform/prebuilts/clang/host/linux-x86/