why and how to use clang compiler with Yocto Project

KHEM RAJ

Yocto Project Summit, 2021.11
Agenda

- Introduction to Clang
- Clang integration
- meta-clang
- Use cases
- Future
- QA
Clang /ˈklæŋ/

- Compiler Frontend, uses LLVM backend
- **Supported Languages**
  - C/C++, Objective-C/C++, OpenMP, OpenCL, RenderScript, CUDA, Fortran
- **Additional compiler analysis tools**
  - Static Analyzer
  - Code analysis tools
- **Uses Apache-2.0 Licence**
Clang /ˈklæŋ/

- **Written in C++**
- **Supported Platform Architectures**
  - Aarch64, ARM, IA-32, x86_64, RISCV32/RISCV64
  - MIPS/MIPS64, PPC64
- **Code**
  - [https://github.com/llvm/llvm-project](https://github.com/llvm/llvm-project)
- **Homepage**
  - [https://clang.llvm.org/](https://clang.llvm.org/)
Clang Goals

- Better Diagnostics
- IDE integration
- Licence compatible with Commercial products
- Developer Friendly
- Fast and low memory use
- GCC compatible
- Modular
Try below samples with Clang & GCC - What do you see?

```
$ gcc -fsyntax-only <input>
$ clang -fsyntax-only <input>
```

```
template<class T>
class a {};
struct b {}
a<int> c;

typedef struct point {
   int x;
   int y;
} point;

struct point origin = { x: 0.0,
y: 0.0 };  
```

```
#include <stdio.h>
int foo() {
   printf("%.*d");
   return 0;
}
```

`attention_to_details.cpp`  
`fixit.c`  
`format.c`
Clang support in Yocto Project

- Support via Independent metadata layer
  - meta-clang - https://github.com/kraj/meta-clang
Using meta-clang

Adding layer

```
$ git clone https://git.yoctoproject.org/git/poky
$ cd poky
$ git clone git://github.com/kraj/meta-clang.git

$ ./oe-init-build-env

$ bitbake-layers add-layer ../meta-clang
```
Using meta-clang

- Default compiler remains gcc
- Choosing clang as compiler
  - Single component - Can be done at recipe level
  - If set in local.conf, it will switch system defaults to clang
- Some packages can not use clang - see conf/nonclangable.conf

```
TOOLCHAIN = "clang"
```
Using meta-clang

- Default runtime remains GNU GCC runtime
- Choosing LLVM runtime
  - LIBCPLUSPLUS - Standard C++ runtime
  - COMPILER_RT - Compiler C runtime
  - UNWINDLIB - System unwinder

RUNTIME = “llvm”
Using meta-clang

● Default C++ Standard Library Switch

LIBCPLUSPLUS = "-stdlib=libc++"

● Per package selection (recipe level)

LIBCPLUSPLUS:toolchain-clang:pn-<recipe> = "-stdlib=libc++"
Using meta-clang

● Generating SDK

CLANGSDK = "1"
Why We need Clang

● Some key packages have switched exclusively to use clang compiler
  ○ Chromium, bcc, bpftrace, SPIRV-LLVM-Translator
    android-tools

● Some prefer clang as first choice
  ○ Firefox, kernel-selftest
Why We need Clang

● Additional Tools
  ○ Static analyser

INHERIT += "scan-build"
SCAN_BUILD ?= ""
SCAN_BUILD:pn-openssl = "1"

$ bitbake -cscanbuild openssl
$ bitake -cscanview openssl
# Why We need Clang

## openssl-3.0.0

### User:
keyup@apoeko

### Working Directory:
/home/yocto/master/build/tmp/work/cortexa72-yoe-linux/openssl/3.0.0-r0/build

### Command Line:
make -j 44

### Clang Version:
clang version 13.0.1 ([https://github.com/facebook/llvm-project/commit/06f0b50c68ff93edee5eb700b50b3a695b00c301534](https://github.com/facebook/llvm-project/commit/06f0b50c68ff93edee5eb700b50b3a695b00c3a695b00c301534))

### Date:
Mon Nov 29 06:22:27 2021

### Bug Summary

<table>
<thead>
<tr>
<th>Bug Type</th>
<th>Quantity</th>
<th>Display?</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Bugs</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Argument with 'nonnull' attribute passed null</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Logic error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assigned value is garbage or undefined</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Dereference of null pointer</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Result of operation is garbage or undefined</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Uninitialized argument value</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Unused code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dead assignment</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>Dead increment</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Dead initialization</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Dead nested assignment</td>
<td>195</td>
<td></td>
</tr>
</tbody>
</table>

### Reports

<table>
<thead>
<tr>
<th>Bug Group</th>
<th>Bug Type</th>
<th>File</th>
<th>Function/Method</th>
<th>Line</th>
<th>Path</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>crypto/param_build.c</td>
<td>param_tsd_convert</td>
<td>333</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>crypto/param_build.c</td>
<td>param_tsd_convert</td>
<td>333</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>crypto/param_build.c</td>
<td>ossl_cms_EncryptedContent_int_bio</td>
<td>165</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>crypto/param_build.c</td>
<td>param_tsd_convert</td>
<td>331</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>test/openssl/conversion_test.c</td>
<td>param_conversion_load_stanza</td>
<td>136</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>test/openssl/conversion_test.c</td>
<td>test_conversion_load_stanza</td>
<td>136</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>test/openssl/conversion_test.c</td>
<td>test_conversion_load_stanza</td>
<td>136</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>test/openssl/conversion_test.c</td>
<td>check_contexts_legacy_PKM</td>
<td>717</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>test/openssl/conversion_test.c</td>
<td>verify_apn</td>
<td>348</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Argument with 'nonnull' attribute passed null</td>
<td>crypto/param_build.c</td>
<td>param_tsd_convert</td>
<td>341</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>Logic error</td>
<td>Assigned value is garbage or undefined</td>
<td>cryto/asn1/asn1_object.c</td>
<td>asn1_ASN1_OBJECT</td>
<td>168</td>
<td>68</td>
<td></td>
</tr>
</tbody>
</table>

*View Report* | Report Bug | Open File
Why We need Clang

● Additional Tools
  ○ LLDB - Debugger
    ■ Provides a target agent llldb-server like gdbserver
    ■ Powerful CLI, python integration
  ○ LLD - Linker
    ■ Very fast linker
    ■ Helps large binaries link times
  ○ Clang-format
  ○ Clangd
  ○ Clang-tidy
Status of Clang compiled packages in Yocto

- **World builds**
  - qemuarm, qemuarm64, qemux86, qemuriscv64, qemuriscv32

- **Distributions**
  - Yoe Distro uses clang as default compiler

- **Packages needing GCC**
  - **Kernel**
    - ARM/x86/RISCV ready to use clang
  - **Glibc**
  - GCC + runtime
  - **Bootloaders e.g. syslinux**
    - u-boot can use clang

- **Complete list**
Limitations

- **Used for target packages only**
  - chromium does use native clang + runtime

- **Relocatable SDKs may not work**
  - Needs special patch to extend ldso paths in binaries

- **Not all target packages will be able to use clang**
  - GCC’ism
  - Dependency on language Undefined behaviors

- **GCC supports many more architectures than Clang**
  - [https://gcc.gnu.org/backends.html](https://gcc.gnu.org/backends.html)
  - Csky, ARC — no luck

- **Reproducible builds are not tested hence may not work**
Wish List

- Use Clang to compile Kernel
- Permanent OE core switch to select compiler
  - TOOLCHAIN variable like
- Yocto Autobuilder Testing
- Benchmarks
- MingW SDK port
- LLVM runtime as default
- List - https://github.com/kraj/meta-clang/issues?q=is%3Aissue+is%3Aopen+label%3Aenhancement
- Enable Extra tools e.g. clang-format etc.
- Enable LTO by default
Thanks for your time!