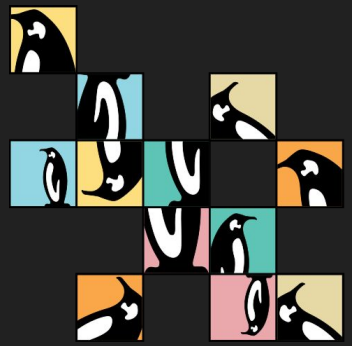
Using clang-tidy and clang-format

Linux Plumbers Conf 2020 - LLVM MC

Miguel Ojeda, Nathan Huckleberry



What is Clang Tooling?

- Clang is one of the LLVM compiler frontends
- But it is also designed as a *platform* for building *source level tools*
  - Syntax checking
  - Automatic fixing of compile errors
  - Code formatting
  - Refactoring
  - Linting
  - Static code analysis
  - …

What could the kernel use Clang Tooling for?

- In the kernel we already use a lot of custom tools
  - `sparse`
  - `spatch` (Coccinelle)
  - `checkpatch.pl`
  - ...
- Clang Tooling allows us to build more of them
  - Without having to deal with parsing C code
  - With support for all C extensions we use
  - With easy access to the AST (Abstract Syntax Tree)

clang-format — What is it?

- A tool to format C-like code
  - Supports different languages
  - Many configurable rules and heuristics
  - Very fast
  - Good enough
- Allows us to spend less time on formatting and reviewing style
- Kernel style already ***pre-configured*** for you
  - Overriding the global style for particular subsystems is possible
  - See `.clang-format`

clang-format — Example

```
static inline void
cpuhp_lock_acquire ( bool bringup ){
 lock_map_acquire ( bringup
   ? & cpuhp_state_up_map
   : & cpuhp_state_down_map );
}
```

clang-format — Example

```
static inline void
cpuhp_lock_acquire ( bool bringup ){
 lock_map_acquire ( bringup
   ? & cpuhp_state_up_map
   : & cpuhp_state_down_map );
}
```

```
static inline void cpuhp_lock_acquire(bool bringup)
{
        lock_map_acquire(bringup ? &cpuhp_state_up_map : &cpuhp_state_down_map);
}
```

clang-format — Common use cases

- Re-formatting blocks
  - After indenting to keep line length within the limit
  - After refactoring and moving code around
- Re-formatting full files
- Sorting `#include`s
- Aligning
  - Variable blocks (on types or on the `=`)
  - Multi-line macro continuations (`\`)
- Reflowing comments
- ...

clang-format — Kernel use cases

- Re-formatting patches
  - Clang provided script: `clang-format-diff.py`
  - Useful to spot coding style mistakes, typos, etc.
  - For both authors and maintainers
- Dealing with “code dumps”
  - e.g. big systems developed out-of-tree and then submitted at once
  - Get them kernel-formatted at a single blow
- Lowering the barrier of entry a tiny bit
  - Kernel development is already different enough
  - One less thing to care about for newcomers

clang-format — How to run it

- In your editor
  - Binding it to a key or at save file time
  - Vim, Emacs, VS Code, CLion, gedit, Sublime Text…
  - See https://clang.llvm.org/docs/ClangFormat.html
- In the command-line
  - `clang-format -i kernel/up.c`
  - `git diff`

clang-format — The {plan,proposal}

- Short-term
  - Get more devs aware of it
  - Get more devs to use it themselves
- Medium-term
  - Get maintainers to format their subsystems with it
  - Converge the slightly different code styles in the kernel
- Long-term
  - Kernel code automatically formatted
  - Style nitpicking in reviews not needed anymore!

clang-format — Further information

- Quick kernel guide
  - `Documentation/process/clang-format.rst`
  - https://www.kernel.org/doc/html/latest/process/clang-format.html
- Official documentation
  - https://clang.llvm.org/docs/ClangFormat.html
  - https://clang.llvm.org/docs/ClangFormatStyleOptions.html

clang-tidy

“Linter” that can identify style violations, interface misuse, or bugs that can be deduced via static analysis

Used to write codebase-specific checks

More than just a linter

Clang-tidy checks

Checks consist of an AST Matcher and a check callback

AST Matcher - Look for AST Node Pattern

Check callback - Run on matching AST nodes

Writing a check

Write AST matcher (https://clang.llvm.org/docs/LibASTMatchersReference.html)

```
auto ErrFn = functionDecl(hasName("ERR_PTR"));
Finder->addMatcher(
    callExpr(callee(ErrFn),hasParent(compoundStmt()))
    .bind("call"),this
);
```

https://clang.llvm.org/extra/clang-tidy/Contributing.html

Writing a check

Write check callback

Matcher gives Clang AST nodes to the check callback

Now we have the full power of Clang’s frontend behind our check

https://clang.llvm.org/extra/clang-tidy/Contributing.html

Running clang-tidy

```
make CC=clang clang-tidy
```

https://patchwork.kernel.org/project/linux-kbuild/list/?series=337007

Clang analyzer

Static analysis tool for finding C/C++ bugs

Can be run through clang-tidy

Path sensitivity

Symbolic execution

Data flow analysis

Running Clang analyzer

Text only mode:

```
make CC=clang clang-analyzer
```

HTML reports

```
scan-build --use-cc=clang make
```

Clang analyzer output

Bug Summary

Bug Type	Quantity	Display?
All Bugs	**1063**	
Dead store		
Dead assignment	382	
Dead increment	76	
Dead initialization	100	
Dead nested assignment	33	
Logic error		
Assigned value is garbage or undefined	52	
Branch condition evaluates to a garbage value	10	
Called function pointer is null (null dereference)	2	
Dereference of null pointer	143	
Dereference of undefined pointer value	4	
Division by zero	22	
Garbage return value	10	
Result of operation is garbage or undefined	77	
Return of address to stack-allocated memory	2	
Uninitialized argument value	72	
Unix API	20	
Memory error		
Bad free	11	☑
Memory leak	4	☑
Use-after-free	43	☑

```
        table = ipv4_net_table;
        if (!net_eq(net, &init_net)) {
                (1) Assuming the condition is false →
        (2) ← Taking false branch →
                int i;

                table = kmemdup(table, sizeof(ipv4_net_table), GFP_KERNEL);
                if (!table)
                        goto err_alloc;

                /* Update the variables to point into the current struct net */
                for (i = 0; i < ARRAY_SIZE(ipv4_net_table) - 1; i++)
                        table[i].data += (void *)net - (void *)&init_net;
        }

        net->ipv4.ipv4_hdr = register_net_sysctl(net, "net/ipv4", table);
        if (!net->ipv4.ipv4_hdr)
                (3) ← Assuming field 'ipv4_hdr' is null →
        (4) ← Taking true branch →
                goto err_reg;
                (5) ← Control jumps to line 1381 →

        net->ipv4.sysctl_local_reserved_ports = kzalloc(65536 / 8, GFP_KERNEL);
        if (!net->ipv4.sysctl_local_reserved_ports)
                goto err_ports;

        return 0;

err_ports:
        unregister_net_sysctl_table(net->ipv4.ipv4_hdr);
err_reg:
        if (!net_eq(net, &init_net))
                (6) ← Assuming the condition is true →
        (7) ← Taking true branch →
                kfree(table);
                (8) ← Argument to kfree() is the address of the global variable 'ipv4_net_table', which is not memory allocated by malloc()
```

Future Work

More checks for the clang-tidy linuxkernel module