

# Intern Project

...

justinstitt@

# Project(s) Overview/Goals



## ClangBuiltLinux:

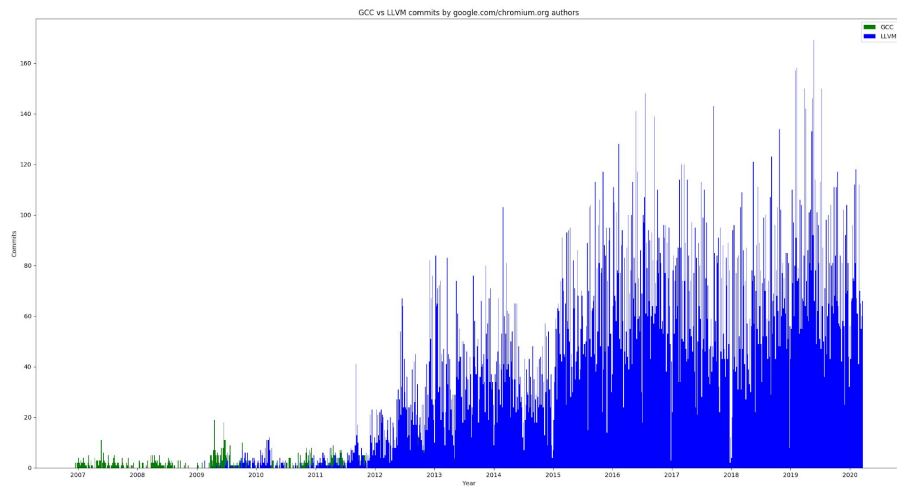
- \* Improve developer trust and ease migration
- \* Fix Clang warnings not caught by GCC



## Clang/LLVM:

- \* Run performance profiles
- \* Identify areas for optimizations
- \* Make the optimizations

# Investing in LLVM and Linux at Google Scale



"Google has large investments in both Linux and LLVM, without ensuring the compatibility of the two. Linux is tightly coupled to a different toolchain, and unless Google invests more in decoupling, the minimum version requirements will soon mean that Google will not be able to upgrade to newer Linux versions, which puts all of Google's products and services at risk. We need to invest in a strong foundation that all of our billion-plus user products rely on."

"Google will no longer have a toolchain capable of compiling the Linux kernel. At this point Google will not be able to upgrade to newer versions of the Linux kernel. For security disasters like Spectre and Meltdown, this will make shipping mitigations more costly."

# ...ClangBuiltLinux - Discrepancies

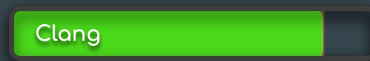
Clang is more strict about certain warnings and therefore catches some that GCC misses. This diverging behavior results in a large amount of warnings built-up over 20+ years to *suddenly* appear.

The **-Wformat** warning is a large offender due to how Clang handles default argument promotion and implicit integer conversion. Over **2,000** of these warnings existed during a previous attempt to enable this warning.

<https://gist.github.com/nathanchance/443db156e56cd3c0f6b21d9d77728d80>

Currently, the warning is **disabled**! This is a poor option as it fails to catch the baseline format warnings that even GCC catches.

-Wformat warnings caught by different compilers



# ...ClangBuiltLinux

```
/* test.c */
#include <stdio.h>

int main() {
    char foo = 4;
    printf("%hhd" "\n", foo + 1);
    return 0;
}
```

## Clang 15

```
~/playground clang test.c -Wall
test.c:5:25: warning: format specifies type 'char' but the argument has type 'int' [-Wformat]
    printf("%hhd" "\n", foo + 1);
               ~~~~ ^~~~~~
                   %d
1 warning generated.
```

## GCC 11.3

```
~/playground gcc test.c -Wall
~/playground ./a.out
> 5
```

# ...ClangBuiltLinux

As previously mentioned, there exists **TONS** of these warnings. Sometimes it is hard to determine what the original developer intended within their *printf-like* function.

For the most part they made a mistake or didn't know about integer promotion rules. Other times they rely on specific behaviors that trigger -Wformat warnings.

```

+ #define NOWARN(option, comment, block) \
+   __diag_push(); \
+   __diag_ignore_all(#option, comment); \
+   block \
+   __diag_pop();
+
KSTM_MODULE_GLOBALS( );

static char *test_buffer __initdata;
@@ -154,9 +160,11 @@ test_number(void)
    test("0x1234abcd ", "%#-12x", 0x1234abcd);
    test(" 0x1234abcd", "%#12x", 0x1234abcd);
    test("0|001| 12|+123| 1234|-123|-1234", "%d|%03d|%3d| %+d| % d| %+d| % d", 0, 1, 12, 123, 1234, -123,
-1234);
-   test("0|1|1|128|255", "%hhu|hhu|hhu|hhu|hhu", 0, 1, 257, 128, -1);
-   test("0|1|1|-128|-1", "%hhd|hhd|hhd|hhd|hhd", 0, 1, 257, 128, -1);
-   test("2015122420151225", "%ho%ho%#ho", 1037, 5282, -11627);
+   NOWARN(-Wformat, "Intentionally test narrowing conversion specifiers.", {
+       test("0|1|1|128|255", "%hhu|hhu|hhu|hhu|hhu", 0, 1, 257, 128, -1);
+       test("0|1|1|-128|-1", "%hhd|hhd|hhd|hhd|hhd", 0, 1, 257, 128, -1);
+       test("2015122420151225", "%ho%ho%#ho", 1037, 5282, -11627);
+   })

```

# ...ClangBuiltLinux - Good News

```
author      Justin Stitt <justinstitt@google.com>      2022-07-20 16:23:32 -0700
committer   Masahiro Yamada <masahiroy@kernel.org> 2022-08-03 19:37:59 +0900
commit      258fafcd0683d9ccfa524129d489948ab3ddc24c (patch)
tree        c99d7caf2372dda8f1f0edc661742cb914362a9b
parent      ee47620367d5b5ee6a1934888bf1ae46576be757 (diff)
download    linux-258fafcd0683d9ccfa524129d489948ab3ddc24c.tar.gz
```

**Makefile.extrawarn: re-enable -Wformat for clang**



<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=21f9c8a13bb2a0c24d9c6b86bc0896542a28c197>  
<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=258fafcd0683d9ccfa524129d489948ab3ddc24c>

my patches: <https://lore.kernel.org/all/?q=justin%20stitt>



# ...ClangBuiltLinux - Bad News

```
author    Linus Torvalds <torvalds@linux-foundation.org> 2022-08-11 08:40:01 -0700
committer Linus Torvalds <torvalds@linux-foundation.org> 2022-08-11 08:40:01 -0700
commit    21f9c8a13bb2a0c24d9c6b86bc0896542a28c197 (patch)
tree      9a7d1edcec7bb8a02dace5115bd35fa9d63204fd
parent    ffcf9c5700e49c0aee42dcha9a12ba21338e8136 (diff)
download  linux-21f9c8a13bb2a0c24d9c6b86bc0896542a28c197.tar.gz
```

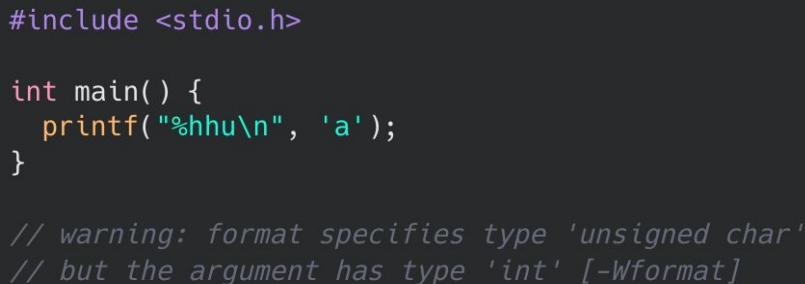
**Revert "Makefile.extrawarn: re-enable -Wformat for clang"**



# Linus' Feedback

"Christ. Why is clang's format warning SO COMPLETELY BROKEN?"

"The clang -Wformat warning is terminally broken, and the clang people can't seem to get their act together."



```
#include <stdio.h>

int main() {
    printf("%hhu\n", 'a');
}

// warning: format specifies type 'unsigned char'
// but the argument has type 'int' [-Wformat]
```

# Linus' Feedback - Possible Solutions

<https://lore.kernel.org/all/C AHk-=wgJA=e-CLcvU5LR Ku0bMLEAewXtOM6as1h FVeQAVkMPbq@mail.gm ail.com/>

## 1) cast

```
#include <stdio.h>

int main() {
    printf("%hhu\n", (char)'a');
}
```

"And no, the "you should use a pointless cast to shut this up" is not a valid answer. That warning should not exist in the first place, or at least be optional with some "-Wformat-me-harder" kind of option"

```
#include <stdio.h>

int main() {
    printf("%hhu\n", 'a');
}

// warning: format specifies type 'unsigned char'
// but the argument has type 'int' [-Wformat]
```

## 2) use correct specifier

```
#include <stdio.h>

int main() {
    printf("%c\n", 'a');
}
```

"truncation is **\*LITERALLY THE MAIN REASON TO EVER USE %hhu IN THE FIRST PLACE\*.**"

# Linus' Feedback (cont.)



```
#include <stdio.h>

int main() {
    printf("%hhx\n", 0xcafe);
    return 0;
}

// warning: format specifies type 'unsigned char'
// but the argument has type 'int' [-Wformat]
// output:
// fe
```



```
#include <stdio.h>

int main() {
    printf("%hhu\n", 256);
    return 0;
}


// warning: format specifies type 'unsigned char'
// but the argument has type 'int' [-Wformat]
// output:
// 0
```

range of %hhu (char) is [0, 255]

# Linus' Feedback - Possible Solutions (cont.)

## 3) implement new warning behavior

### -Wformat-me-harder (TBD)



```
#include <stdio.h>

int main() {
    printf("%hhu\n", 256);
    return 0;
}

// warning: format specifies type 'unsigned char'
// but the argument has type 'int' [-Wformat-me-harder(TBD)]
// output:
// 0
```

"And until clang fixes their idiotic warning policy, that -Wformat stays turned off.

Now, if you can show that the clang people understand **why that warning is completely bogus and broken, and they've fixed it in their development tree**, at that point I'm willing to turn the warning on again and work around it in the kernel."

# Clang/LLVM

**Problem:** As it stands building the Linux Kernel with Clang is slower than with GCC.

**Reasoning:** "I really wish clang wasn't so much noticeably slower. It's limiting what I do with it, and I've had other developers say the same." - Linus Torvalds

**Solution:** Make builds faster by running profiles and implementing optimizations to the compiler itself.



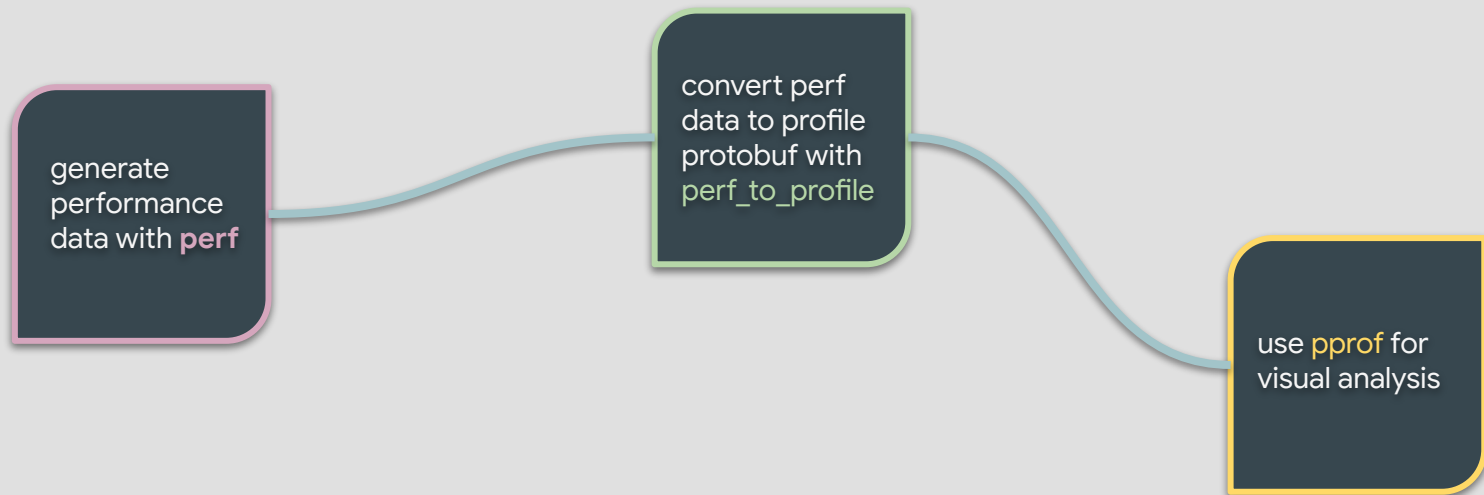
Linus Torvalds - TED  
([https://www.ted.com/talks/linus\\_torvalds\\_the\\_mind\\_behind\\_linux](https://www.ted.com/talks/linus_torvalds_the_mind_behind_linux))

# ...Clang/LLVM

## Let's talk about Perf.



# ...Clang/LLVM - **perf** to **pprof**





# pprof (demo) - <https://github.com/google/pprof>

A tool to visualize performance profiles.



<https://github.com/ClangBuiltLinux/profiling/tree/main/pprof>

# ...pprof - bottom-up view

SELF	%	LOCATION
197,542,739,308	2.474%	▸ clang::SourceManager::getFileIDLocal < getFileID
160,618,144,827	2.012%	▸ llvm::StringMapImpl::LookupBucketFor
107,747,918,147	1.350%	▸ clang::TokenLexer::Lex
97,141,301,534	1.217%	▸ (unknown)
92,634,636,255	1.160%	▸ clang::Lexer::LexTokenInternal
80,226,465,782	1.005%	▸ _int_malloc (malloc.c)
78,174,170,098	0.979%	▸ (anonymous namespace)::IntExprEvaluator::VisitBinaryOper
74,680,688,483	0.935%	▸ clang::Type::getAs
72,454,240,795	0.908%	▸ clang::Lexer::LexIdentifierContinue
69,513,520,398	0.871%	▸ GetDiagInfo (DiagnosticIDs.cpp)
67,011,277,692	0.839%	▸ clang::Preprocessor::Lex
64,420,592,888	0.807%	▸ clang::ASTContext::getDeclAttrs
64,116,684,581	0.803%	▸ CheckICE (ExprConstant.cpp) < CheckICE
53,126,699,503	0.665%	▸ GetFullTypeForDeclarator (SemaType.cpp)
50,424,315,453	0.632%	▸ clang::TokenLexer::ExpandFunctionArguments
47,380,616,032	0.594%	▸ (unknown) (memmove-vec-unaligned-erms.S)
45,807,512,107	0.574%	▸ clang::SourceManager::isOffsetInFileID (Diagnostic.cpp)

Good perspective on hot methods.

<https://github.com/ClangBuiltLinux/profiling/tree/main/pprof>

# DiagnoseNullConversion

## What is a null conversion?

**C std:** "The macros are NULL which expands to an implementation-defined null pointer constant"

**C++ std:** "Possible definitions include 0 and 0L, but not (void\*)0"

```

    ● ● ●
#ifdef __cplusplus
# if !defined(__MINGW32__) && !defined(_MSC_VER)
#   define NULL __null
# else
#   define NULL 0
# endif
#else
# define NULL ((void*)0)
#endif
```

../clang/stddef.h +86

```

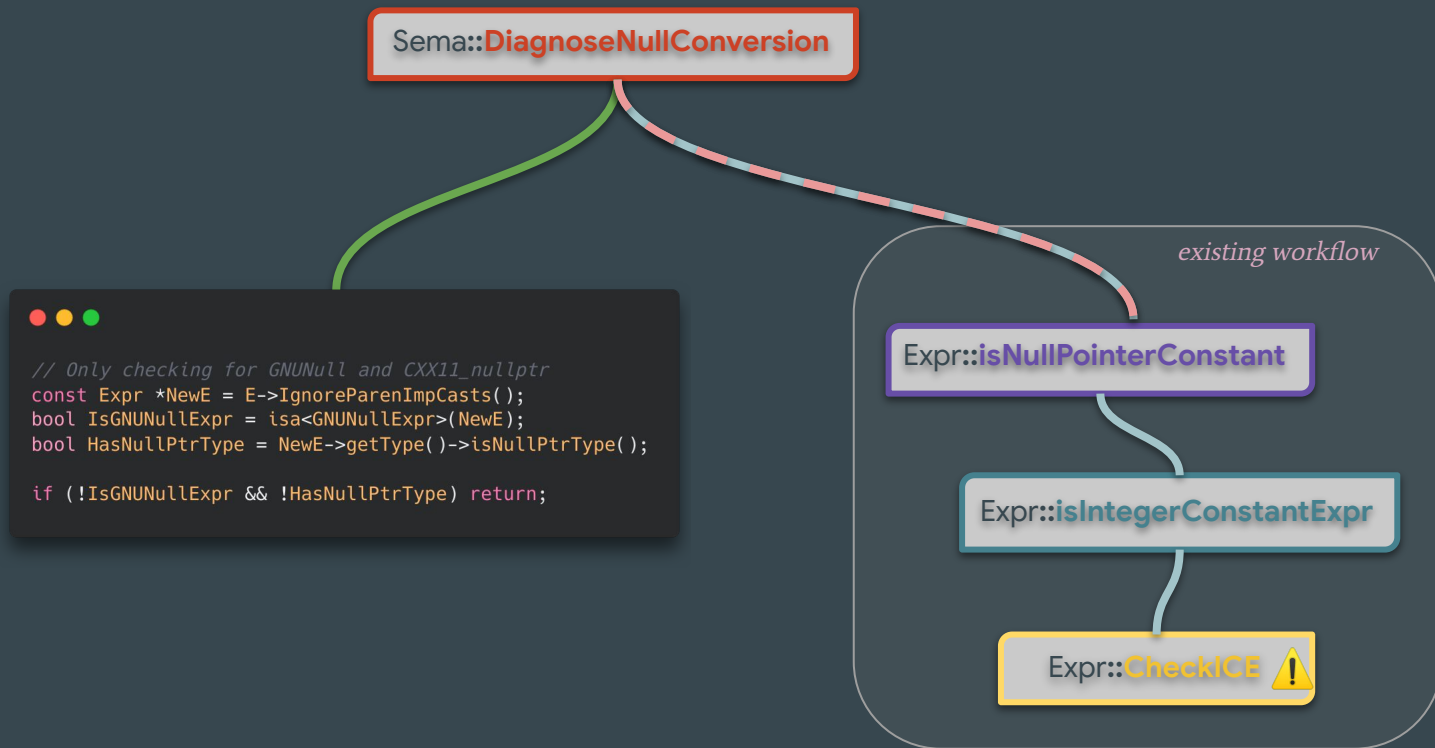
    ● ● ●

int main() {
    int a = NULL;
    return 0;
}

// clang:
// warning: implicit conversion of
// NULL constant to 'int' [-Wnull-conversion]

// GCC:
// warning: converting to non-pointer
// type 'int' from NULL [-Wconversion-null]
```

# DiagnoseNullConversion (cont.)



# Measurable Gains

wall time builds:

<u>x86_64 clang vanilla:</u> n=10	<u>with patch:</u> n=10
<ul style="list-style-type: none"><li>• 1:16.73</li><li>• 1:17.27</li><li>• 1:17.35</li><li>• 1:17.21</li><li>• 1:17.41</li><li>• 1:16.97</li><li>• 1:17.19</li><li>• 1:17.22</li><li>• 1:17.52</li><li>• 1:17.32</li></ul>	<ul style="list-style-type: none"><li>• 1:15.34</li><li>• 1:15.52</li><li>• 1:15.58</li><li>• 1:15.63</li><li>• 1:15.59</li><li>• 1:16.05</li><li>• 1:15.50</li><li>• 1:15.73</li><li>• 1:15.67</li><li>• 1:15.58</li></ul>
mean=77.21900000000001 seconds.	mean=75.619 seconds.


~2.1% total speed increase.


SELF	%	LOCATION
61,567,476,698	0.799%	▸ <b>CheckICE</b> (ExprConstant.cpp)


SELF	%	LOCATION
4,272,352,217	0.053%	▸ <b>CheckICE</b> (ExprConstant.cpp)


<https://reviews.llvm.org/D131532>

# Measurable Gains (cont.)

 **[Sema] Avoid isNullPointerConstant invocation**

 Closed

 Public

 Authored by **justinstitt** on Tue, Aug 9, 5:00 PM.

# W.I.P: [specifics about potential Clang improvements]

## getFileID

- \* caching
- \* smarter search
- \* MacroArgExpansion
- \* AOS  $\rightarrow$  SOA

## CheckICE

- \* caching (Expr literals, Expr\*)
- \* Sub-expression short circuiting

**Thanks!**

Any  
Questions?

—