Google

2026 EURO LLVM

Object Size Reduction

via shortening STRTAB entries

EuroLLVM
April 2026

Authors: **rnk**@,**vyng**@, et al
Speaker: **vyng**@google.com

Motivation

The Problem

- Large C++ object files are a major cost to Google build systems.
  - Binaries reach 2.5GB+
  - The **STRTAB** accounts for **~35%** of total size.
    - ~5.5m symbols in one single binary
    - Max symbol length ~760k
    - ~6 trillion total symbols

Optimization opportunities!

Glossary

SYMTAB

Symbol table. Contains addresses + offsets into the STRTAB rather than actual names

STRTAB

String table. Flat array of null-terminated strings; stores names for the symbol table. (SYMTAB)

DYNSTR

Dynamic version of STRTAB; required for dynamic symbols and cannot be stripped.

Premise

Usages for strings in STRTAB

- **can** be used for debugging/symbolization(primarily on-task, when DWARF is unavailable)
- match references to definitions

But their specific text is **arbitrary**; **original strings not required.**

Shortening these names shrinks the STRTAB without breaking the builds. 

"What's in a name?"

- William Shakespeare

(*Linker*: "That which we call a symbol;
By any other name would still have the same adress")

Solution

Symbols renaming

Option 1: New mangling scheme

Modify the C++ name mangler to generate shorter mangled names.

Option 2: STRTAB Hashing

Apply transformations on the StringTable entries before emitting the section.

Solution: new mangling scheme

Goal: Keep only the scope name to reduce identifier length.

ORIGINAL NAME

Demangled:

```
T<A> a::b::c::MyFunc<B>
(P<C1>,P<C2>,P<C3>,P<C4>,P<C5>, ..., P<C8>)
```

Mangled:

```
_ZN1a1b1c6MyFuncI1BEE1T1AI1EEP1C1IS3_EP1C2I
S5_EP1C3IS7_EP1C4IS9_EP1C5ISB_EP1C6ISD_EP1C7
ISF_EP1C8ISH_E (101)
```

NAME IN NEW SCHEME

Demangled:

```
a::b::c::MyFunc_1942006764724b75
```

Mangled:

```
_ZN1a1b1c6MyFuncEv_161942006764724b75V (36)
```

** Removes template parameters and return types while preserving hierarchical scope.*

Solution: STRTAB Hashing

Goal: Hash C++ mangled names into a fixed-length format.

- `_<hash><sentinel>`
  - A single-character prefix (_)
  - A 16-hex character hash (BLAKE3)
  - A 2-character sentinel string (eg., `"\x1\x2"`)
- **Length:** fixed at 19 chars.

Solution: STRTAB Hashing

ORIGINAL NAME

```
T<A> a::b::c::MyFunc<B>
(P<C1>,P<C2>,P<C3>,P<C4>,P<C5>,..., P<C8>)

_ZN1a1b1c6MyFuncI1BEE1T1AI1EEP1C1IS3_EP1C2I
S5_EP1C3IS7_EP1C4IS9_EP1C5ISB_EP1C6ISD_EP1C
7ISF_EP1C8ISH_E (101)
```

NEW SCHEME

```
_D24BA381F7E8BAD3\x1\x2(19)
```

Impact

STRTAB size

~85%

reduction

Linker input (without FDO or debuginfo)

15%-25%

reduction

Linker input (with FDO+debuginfo)

~10%

reduction

Output Binary size

10%-25%

reduction

FAQs

Hash collisions (64-bit hash)

$$1 - \frac{(2^{64})!}{(2^{64})^k (2^{64} - k)!}$$

- P(5.5mils) = ~0.000082%
- "Danger-zone" (50%+) at 4.3B symbols
- Collision checking in place

Why not strip the whole strtab?

Negatively affect online symbolizations.

Will it be upstreamed?

Maybe.

Error handling (linker errors)

`.zstrtab`, which contains compressed set of original symbols for look up in case of errors.

Symbolization impact

- **With debuginfo:** No impact; original names stored in `DW_AT_linkage_name`
- **W/O debuginfo:** Requires referencing `.zstrtab`