

Towards real-time ABI compatibility assurance and smarter module versioning with `libctf` or, yet another attempt at better module versioning

Nick Alcock

October 4, 2026

Maybe this is all redundant



All based on BTF... with additions...

- ▶ GNU ld will contain a BTF deduplicator once it goes upstream
- ▶ the actual BTF is generated by GCC, but I hope to add LLVM support fairly soon if nobody else does
- ▶ based on `libctf`, of course, an evolution of the CTF deduplicator it already has
- ▶ we can revive it (particularly now that ID assignment has moved out of `pahole`), and deduplicate compiler-generated BTF directly, without needing DWARF
- ▶ deduplicators need to tell if two types are identical to work at all: we use recursive hashing, as does `gendwarfksyms`
- ▶ can we repurpose this for DWARFless extended modversions?
- ▶ all up in the air so far, nothing implemented, all guesswork

The (slight) trouble with modversions

- ▶ `gendwarfksyms` traverses the type graph for each type, chains types across TUs, turns the full description into a long string (largely consisting of type names), and hashes it, modified by `gendwarfksyms` rules
- ▶ that's what `libctf` used to do too, long ago (except we mixed in DWARF file and line numbers as well). It... sort of works.
- ▶ it intrinsically assumes that types with the same name are the same type. This is absolutely not required by C (though LTO does want it).

The (slight) trouble with modversions

- ▶ I personally ran into a dozen such clashes. There are still some, even though they tend to break things when BTF usage hits them and they do get fixed. On at least one occasion I know of they broke things so badly that all the BTF selftests failed.
- ▶ `libctf`'s algorithm doesn't have this problem.
- ▶ But this is marginal: I am mostly motivated by the desire to get modversions without needing DWARF at all.
- ▶ This might, after all, be redundant!

The libctf algorithm

- ▶ covered in too many previous talks, but in brief...
- ▶ types reduced to hashes directly, with no string intermediate, with named entities tied to a name so we can detect clashes
- ▶ only structs with tags are encoded as a name ("s foo"), eliminating cycles in the type graph before they can happen
- ▶ parallelizable in theory: will finish that as I work on this (the work is about half done, thanks Bruce!)

The general approach

- ▶ export the libctf deduplicator's currently private hashing machinery, so that external tools can see the stream of hashes (and ask to generate it without doing a full-blown `ctf_link()`, type emission and all)
- ▶ the hashes are not quite what the deduplicator uses; we use the *symtypehashes* described in my recent Cauldron talk: <https://www.esperi.org.uk/~nix/ctf/2026-cauldron.pdf>; they include only enough information to describe ABI concerns, not API ones (no names, no enumerands, etc, just type kinds and sizes really).
- ▶ add enough configurability to let us use `gendwarfksyms` rules unmodified
- ▶ write an external tool that uses this to make modversions hashes and put it in `scripts/genbtfsyms`.

How to implement gendwarfksyms rules?

- ▶ maybe allow the hasher to take a pile of *overriding* BTF (in `libctf` parlance, a *dict*), and replace types we see in the hashing process with ones with the same name in the overriding dict. But that only works for types with names...
- ▶ or we could use BTF tags! if we encode all the rules into BTF tags, we can have the hashing machinery call `genbtfksyms` back when it encounters one, and it can do whatever it wants to apply the tag. This feels much more elegant.
- ▶ probably it should take the type that would have been hashed and return *another* type that should be hashed instead, or `void (0)`.

How to implement gendwarfsyms rules?

- ▶ we may also want to avoid hashing non-public subtypes? I guess this is already covered by gendwarfsyms rules though.
- ▶ we definitely only want to hash those top-level types that are `EXPORT_SYMBOL*`
- ▶ we may *possibly* also want a way to say ‘if you see this type, don’t hash anything that includes it’ (a poison pill for things that should never get into any public interfaces): might cut down some of the existing gendwarfsyms tags, possibly?

Putting it all together

- ▶ once this is done we have a pile of hashes for each TU
- ▶ the deduplicator can already turn this into a two-level tree, with the top level containing hashes with unambiguous definitions, and the second level containing the ambiguous ones.
- ▶ that top level is probably what modversions wants. (If we ever find ourselves relying on an ambiguous type in the module API this will tell us immediately so we can fix it: they all show up as forwards, though I'm tempted to add BTF tags to all of them to make it clearer.)

Caveats

- ▶ None that I know of, which means I haven't thought about this enough yet.

Thanks

- ▶ Elena Zannoni for API improvements, countless suggestions
- ▶ Indu Bhagat, Jose Marchesi and David Faust for the BTF and CTFv3 generators in GCC (I'll do the v4 one) and for lots of useful critiques
- ▶ Bruce McCulloch for the first cut at a pahole + libctf-based BTF deduplicator

Links

- ▶ libctf rewrite branch in progress (close to submission):
<https://sourceware.org/git/?p=binutils-gdb.git;a=shortlog;h=refs/heads/users/nalcock/road-to-ctfv4>
- ▶ Cauldron talk slides (video coming soon): <https://www.esperi.org.uk/~nix/ctf/2026-cauldron.pdf>