GCC BPF update

GCC BPF team

LSFMMBPF 2026

Outline

Update on BPF support in GNU Toolchain

BPF Compiler Support

Some discussion topics

Update on BPF support in GNU Toolchain

BPF Compiler Support

Some discussion topics

The tools

GCC

- bpf-unknown-none-gcc

GNU binutils, **GDB** and the **GNU simulator**

- bpf-unknown-none-{as,objdump,...}
- gdb
- bpf-unknown-none-run

DejaGNU

- bpf-sim.exp, bpf-vmtest.exp

GNU poke

- bpf.pk, bpfcore.pk
- btf.pk, btf-dump.pk, btf-ext.pk

Elfutils

- eu-{readelf,objdump,...}

General updates

- **GCC 16.1** released last week!
- Team is growing: welcome Vineet Gupta!
- BPF specific **mailing list**: bpf@gcc.gnu.org.
- https://gcc.gnu.org/wiki
  - BPFBackEnd
  - BPFBackEnd/ABI
  - BPFRunTimeTests
- Weekly **meetings** over BBB.
  - Every Monday from 19:00 to 20:00 CET/CEST.
  - https://bbb-new.sfconservancy.org/rooms/wgy-wvm-vyt-id4/join
- Kernel selftests status: 601/5488 PASSED, 126 SKIPPED, 112 FAILED

binutils updates

- **wN** regs are now allowed in load/store instructions.
- Support for **may_goto** instruction.
- BTF support in objdump via libctf (WIP).
- Solana BPF (sBPF) ISA support (WIP).
  - `github.com/solana-foundation/solana-improvement-documents/`

sBPF ISAs

Feature	**v0**	**v1**	**v2**	**v3**
Static syscalls	no	no	no	yes
Dynamic stack frames	no	yes	yes	no
PQR instructions	no	no	yes	no
Explicit sign extension	no	no	yes	yes
Compatibility with eBPF v3 ISA	no	no	no	yes

Bugzillas

28565	gdb BPF support . gdb crashes after step/run/continue

GCC updates (I)

- GCC now generates **line info** in BTF.ext.
- **ABI** consolidation and fixes.
- **Code generation** fixes
  - Zero extension now uses 32-bit instructions.
  - **BYTE_SLOW_ACCESS** is now zero.
  - **memmove** and **memset** inlining and miscompilations.
- **CO-RE** fixes.
  - More about this later...
- CO-RE handling of bit fields in structs.
- New BPF torture testsuite, bpf-vmtest-tool
- Push and pop attribute pragmas. (WIP)

GCC updates (II)

Bugzillas

121419	__clzdi2 emitted when building udev-hid-bpf
124974	bpf: support for bpf arenas
124698	bpf: support stack arguments
124419	adjust `PROMOTE_MODE` and sign/zero extension
124171	procedure calling convention ABI incompatible with clang
123516	xdp-tools fails tests for bpf
121461	switch to table conversion should be switched off for BPF
119941	bpf: `-Warray-bounds` warning in `adjust_offset_for_component_ref`
117068	bpf: dereference of modified ctx ptr is disallowed
116718	bpf: support `bpf_fastcall` attributes
107438	bpf: support passing small record arguments by value

GCC updates (III)

BPF torture test, expected to verify.

```
1   // { dg-do run  }
2   // { dg-options -Wall }
3
4   #include "vmlinux.h"
5   #include <bpf/bpf_helpers.h>
6   #include <bpf/bpf_tracing.h>
7   #include <bpf/bpf_core_read.h>
8
9   char LICENSE[] SEC("license") = "GPL";
10
11  int example_pid = 0;
12
13  SEC("tracepoint/syscalls/sys_enter_openat")
14  int handle_openat(struct trace_event_raw_sys_enter *ctx)
15  {
16          int pid = bpf_get_current_pid_tgid() >> 32;
17          char filename[256]; // filename buffer
18          bpf_probe_read_user(&filename, sizeof(filename), (void *)ctx->args[1]);
19          bpf_printk("sys_enter_openat() called from PID %d for file: %s\n", pid, filename);
20
21          return 0;
22  }
23
24  /* { dg-output "BPF programs successfully loaded" } */
```

GCC updates (IV)

BPF torture test, expected to not verify.

```
1   // { dg-do run { xfail *-*-* } }
2   // { dg-options -Wall }
3
4   #include "vmlinux.h"
5   #include <bpf/bpf_helpers.h>
6   #include <bpf/bpf_tracing.h>
7
8   char LICENSE[] SEC("license") = "GPL";
9
10  SEC("tracepoint/syscalls/sys_enter_openat")
11  int bpf_prog(struct trace_event_raw_sys_enter *ctx) {
12      int arr[4] = {1, 2, 3, 4};
13
14      // Invalid memory access: out-of-bounds
15      int val = arr[5];  // This causes the verifier to fail
16
17      return val;
18  }
```

DejaGNU updates

- baseboards/bpf-vmtest.exp by Piyush Raj.
- Repo git://git.sv.gnu.org/dejagnu.git.
- Currently in branch feature-bpfvmtest, soon in master.
- baseboards/bpf-sim.exp also available but not upstream.

elfutils updates

Bugzillas

30628	bpf: relocations are outdated in elfutils
30629	bpf: complete BPF disassebler in elfutils

Update on BPF support in GNU Toolchain

BPF Compiler Support

Some discussion topics

BPF Compiler Support

As of May 2026.

Feature		LLVM	GNU
ISA v1-v4	production	yes	yes
ALU-32	production	yes	yes
lineinfo in BTF.ext	production	yes	yes
CO-RE relocs in BTF.ext	production	yes	yes
CO-RE preserve_static_offset	production	yes	patch
Other CO-RE attributes	production	yes	yes
Memory ordering aware atomic built-ins	production	yes	no
attribute push/pop pragmas	production	yes	no
small record arg passing	production	yes	no
fast calls	production	yes	no
may_goto instruction	production	yes	yes
addr_space attribute	experimental	yes	no
addr_space_cast insn	experimental	yes	no
stack args	proposed	patch	no
-fverifiable	proposed	no	no

What is not in the table?

Update on BPF support in GNU Toolchain

BPF Compiler Support

Some discussion topics

CO-RE related topics

- Split CO-RE and non CO-RE expressions.
  - Some memory accesses involve both CO-RE and non-core structs.
- Attribute push/pop pragmas in GCC.
  - bpftool uses it in generated vmlinux.h.
  - Workaround (in GCC) for C not having nested structs.
- Bit fields in CO-RE.
  - GCC now generates correct code like clang.
  - But they don’t generate CO-RE relocs for these fields.
  - Worth supporting them? If no, should we warning?
- CO-RE and packed structs.
  - Something seems to be off with packed structs and CO-RE.
  - `https://godbolt.org/z/eh4ehWKTo`
- CO-RE attribute **preserve_static_offset**.
  - Clang compiled code doesn’t seem to need this as of today.
  - But GCC does.

BTF related topics

- We finally have **type tags** and **decl tags** in GCC, both DWARF and BTF.
- GCC now matches clang in the ordering of type tags in pointer types.
- BTF and function signature changes in optimized code.
  - Alan and Yonghong will talk about this later.
  - Can this be reliably conveyed in DWARF?
  - GCC can generate BTF for any target.

Other topics

- Implement some compiler buil-ins like __builtin_clz as kfuncs.
  - Does this make sense?
  - With fast calls, these are fast!
- Linking BPF programs
  - ELF link editor, BPF loader.
  - GNU ld can link BPF programs, limited by existing relocs.
  - Relocs, dynamic tags.
  - What are the plans?
- Arguments on the stack
  - Proposal by Yonghong. Discussed later.

And more!

- Implement cost model is the BPF backend.
  - This helps to generate more verifiable code.
- Argument promotion in GCC should match clang.
- More work on the verifier is necessary to handle GCC codegen idioms.

Thanks

Let’s catch up during the conference to talk about these and other topics... Thank you!