



# BPF Support in the GNU Toolchain

GCC BPF team

LPC 2026 | Oct. 5-7 | Prague

# Outline

State of the Port

BPF Compiler Support

Open Items and Discussion Topics

## State of the Port

BPF Compiler Support

Open Items and Discussion Topics

# The Tools

## **GCC**

- ▶ bpf-unknown-none-gcc

## **GNU binutils and GDB**

- ▶ bpf-unknown-none-{as, objdump, ...}

## **DejaGNU**

- ▶ bpf-sim.exp, bpf-vmtest.exp

## **GNU Poke**

- ▶ bpf.pk, bpfc.core.pk, pk-bpfc.core.dump
- ▶ btf.pk, btf-ext.pk, btf-dump.pk

## **Elfutils**

- ▶ eu-{readelf,objdump,...}

# The Team

- ▶ <https://gcc.gnu.org/wiki/BPFBackEnd>
- ▶ [bpf@gcc.gnu.org](mailto:bpf@gcc.gnu.org)
- ▶ #gccbpf on irc.oftc.net
- ▶ Weekly meetings on BBB: Mondays at 1900 CET/CEST
- ▶ Welcome Vineet Gupta
- ▶ Contributions welcome!
  - Try compiling your project with GCC BPF and send bug reports :)

## Distros

- ▶ Debian and derivatives
- ▶ Fedora and derivatives
- ▶ Gentoo and derivatives
- ▶ Oracle Linux
- ▶ ...

## Programs

- ▶ systemd (BPF components)
- ▶ bpftune
- ▶ dtrace
- ▶ ...

# GCC Updates

## Bugzillas

|               |                                                                                          |
|---------------|------------------------------------------------------------------------------------------|
| <b>121419</b> | <code>__clzdi2</code> emitted when building <code>udev-hid-bpf</code>                    |
| <b>124974</b> | bpf: support for bpf arenas                                                              |
| <b>124698</b> | bpf: support stack arguments                                                             |
| <b>124419</b> | adjust <code>PROMOTE_MODE</code> and sign/zero extension                                 |
| <b>124171</b> | procedure calling convention ABI incompatible with clang                                 |
| <b>123516</b> | xdp-tools fails tests for bpf                                                            |
| <b>121461</b> | switch to table conversion should be switched off for BPF                                |
| <b>119941</b> | bpf: <code>-Warray-bounds</code> warning in <code>adjust_offset_for_component_ref</code> |
| <b>117068</b> | bpf: dereference of modified ctx ptr is disallowed                                       |
| <b>116718</b> | bpf: support <code>bpf_fastcall</code> attributes                                        |
| <b>107438</b> | bpf: support passing small record arguments by value                                     |
| <b>126061</b> | bpf: support returning small record arguments by value                                   |

# BPF vmtest tool

Contributed by GSoC 2025 student Piyush Raj

- ▶ `gcc/contrib/bpf-vmtest-tool/`
- ▶ Scripts to hook kernel vmtest tooling to DejaGNU for GCC testsuite
- ▶ Fetch and build specified kernel version
- ▶ Compile test programs and load them
- ▶ Capture verifier logs and check with DejaGNU directives

# BPF vmtest tool

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" } */
```

# BPF vmtest

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 }
```

# CO-RE Updates

- ▶ Splitting expressions into CO-RE and non-CO-RE parts
- ▶ CO-RE and bit fields
  - Clang and GCC produce code that is not always patchable (WIP)
- ▶ CO-RE and packed structs
  - Similar issue to bit fields
  - <https://godbolt.org/z/eh4ehWKTo>
- ▶ `#pragma attribute push/pop`
  - Used by bpftool in generated `vmlinux.h`, will be phased out soon
  - But general interest from GCC side in supporting the pragmas

## Bit-op built-ins via kfuncs

- ▶ PR121419: `__clzdi2` emitted when building `udev-hid-bpf`
- ▶ GCC prefers to implement these with libcalls
- ▶ Same problem for `ctz`, `ffs`, `popcount`, ...
- ▶ Expanding to BPF bytecode precludes using dedicated insns when available
  - most supported architectures have a dedicated `clz` instruction
  - general `__clzdi2` requires 31 BPF instructions
- ▶ Patches to `bpf-next` from Leon Hwang adding bitops kfuncs
  - Allows arch-specific JITs to manually expand the ops
  - Once the feature lands, perfect for implementing compiler built-ins

State of the Port

**BPF Compiler Support**

Open Items and Discussion Topics

# BPF Compiler Support

As of Oct 2026.

| Feature                                |              | LLVM  | GNU   |
|----------------------------------------|--------------|-------|-------|
| ISA v1-v4                              | production   | yes   | yes   |
| BTF.ext funcinfo,lineinfo              | production   | yes   | yes   |
| CO-RE support                          | production   | yes   | yes   |
| CO-RE preserve_static_offset           | production   | yes   | patch |
| Other CO-RE attributes                 | production   | yes   | yes   |
| BTF type_tag and decl_tag              | production   | yes   | yes   |
| Memory-ordering aware atomic built-ins | production   | yes   | no    |
| small record arg/retval 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   | patch |
| indirect calls                         | experimental | yes   | patch |
| stack args                             | proposed     | patch | no    |
| -fverifiable                           | proposed     | no    | no    |

State of the Port

BPF Compiler Support

Open Items and Discussion Topics

# Rust

- ▶ GCC Rust front-end
- ▶ `rustc`  $\rightarrow$  LLVM [ $\rightarrow$  DWARF]  $\rightarrow$  BTF
- ▶ `gccrs` [ $\rightarrow$  DWARF]  $\rightarrow$  BTF
  - Are they compatible?
  - Are they composable?
- ▶ Is DWARF produced by `rustc` and `gccrs` intelligible to pahole?
- ▶ Is compiler-emitted BTF for Rust-source BPF progs correct?
- ▶ Alan Maguire's talk: extending BTF to non-C languages
- ▶ Michal Rostecki: CO-RE support for Rust - 1pm Wednesday

# BPF Verifier and GCC

- ▶ BPF verifier evolved with LLVM backend and LLVM-isms
- ▶ Some GCC codegen idioms are not well understood by the verifier even though they are sound
  - e.g. reusing a register known to contain a NULL pointer
- ▶ Testsuite hygiene / workflow updates
  - Better skip reporting
- ▶ Teaching verifier about sign extension
  - GCC codegen uses sign extensions much more
- ▶ Dedicated talk: 3pm in BPF track

## Other Topics

- ▶ Indirect calls (`callx %rN`)
- ▶ `-mcpu=experimental?`
- ▶ Using instruction cost model for verifiability
- ▶ Argument promotion in GCC vs. clang
- ▶ Solana BPF