Standard Documentation

How to scope and organize the effort under the eBPF Foundation

Background

- Dave Tucker's "eBPF Documentation" thread from October 2021
  - https://lore.kernel.org/bpf/CAOJ0YmrUNbw_qMP_FHmoYejS1JRaKCkD69S5xYS9gxsWAPX4rw@mail.gmail.com/
- Christoph Hellwig's "LSF/MM session: eBPF standardization"
  - https://lore.kernel.org/bpf/20220503140449.GA22470@lst.de/
- Jim Harris:
  - NVMe is developing a Technical Proposal (TP4091) for computational storage
  - TP4091 would like to define how a host can download a program to an NVMe device that is expressed with eBPF
  - Lack of a formal eBPF standard is a bit of sticking point however

What exists, some examples

- Linux man pages
  - List of prog types, map types, helper function docs
- https://github.com/iovisor/bpf-docs (but not actively maintained)
  - **bpf_helpers.rst**: docs on helper functions
  - **eBPF.md**: Instruction Set Architecture (ISA) for eBPF VM
- Linux kernel.org sources
  - filter.txt: “official documentation for the eBPF instruction set” per bpf-docs
- https://github.com/torvalds/linux/tree/master/Documentation/bpf
  - btf.rst: BTF Type Format
  - verifier.rst: Verifier expectations
- https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md
  - List of BPF features by Linux kernel version
  - List of License requirements per helper function
  - List of helper functions per prog type
- A couple of other sites listed in Dave Tucker’s mail

Gaps

- No central place to look, so hard to find
- Docs may or may not be up to date
- Examples are spotty
- Some ISA expectations not well documented
  - Underflow, overflow, divide by zero, etc.

Desired State?

- Dave Tucker's proposal:
  1. ebpf.~~io~~*foundation* is the home to all official documentation
  2. Documentation arranged from bottom of the stack up
     1. eBPF VM Instruction Set
     2. ELF File Format, BTF and CO:RE
     3. eBPF compilers
     4. eBPF Syscall Interface
     5. eBPF Program Types, Map Types and Helpers
  3. Docs that then point to the API documentation for libbpf, libxdp, bcc, and various other libraries.

Possible principles for scoping/framing

- Distinguish between cross-platform vs runtime (or even runtime-version specific) info
  - Should runtime-specific stuff be in scope or out of scope for Foundation-maintained standard docs?
  - There's a difference between something that is inherently runtime-specific vs something that is just on one runtime so far… when does it become "standard"?
- Do we use must/should/may language regarding runtimes? (yes)
- Do we have mandatory/conditionally-mandatory/optional features? (use above)
- Do we have multiple versions? (yes)
- How do we handle deprecated features or APIs?
- Do we have a compliance check mechanism somehow? (not common yet?)
  - Compliance of an ebpf program, vs compliance of a runtime

What components might be in scope?

- DThaler proposes constraining initial scope (but maybe more later):
  - IN SCOPE: Things that directly affect ebpf program source, bytecode, or toolchains
    - ISA, ELF format, BTF (load time vs debug/metadata info), some verifier expectations in terms of what ISA covers vs verifier (not isa responsibility), compiler expectations
  - OUT OF SCOPE:
    - prog types, map types, verifier expectations (test for halting), macros like SEC(), etc.
  - OUT OF SCOPE: platform-specific (and compiler-specific?) details
    - Actual tracepoints, platform-specific helper functions, etc.
  - OUT OF SCOPE: specific APIs for how apps interact with ebpf programs
    - Libbpf, syscalls, etc.
  - OUT OF SCOPE: specific tools for how users interact with ebpf programs
    - Bpftool, bcc, etc.
  - But **ok to point to other docs for things out of scope**
- Other views?
  - With more effort we could try to cover all production platforms, but do we really want to?

Organizing the effort

- License/copyright:
  - Can we actually take text from other sites?
  - Bcc-docs has no license info that I could find (bcc is Apache)
- Repo:
  - new repo under https://github.com/ebpffoundation?
  - fork of bpf-docs?
- Meetings:
  - None, just PRs?
  - Ad hoc meetings as needed, during bpf office hours slot? ←this one
  - Periodic meetings during a different slot?
- Volunteers:
  - Split up doc categories among volunteers to do a first cut?
  - Who is willing?