

Name

bpftool-iter – tool to create BPF iterators

SYNOPSIS

bpftool [*OPTIONS*] **iter** *COMMAND*

OPTIONS := { { **-j** | **--json** } [{ **-p** | **--pretty** }] | { **-d** | **--debug** } }

COMMANDS := { **pin** | **help** }

ITER COMMANDS

bpftool iter pin *OBJ PATH* [**map** *MAP*]

bpftool iter help

OBJ := /a/file/of/bpf_iter_target.o

MAP := { **id** *MAP_ID* | **pinned** *FILE* }

DESCRIPTION

bpftool iter pin *OBJ PATH* [**map** *MAP*]

A bpf iterator combines a kernel iterating of particular kernel data (e.g., tasks, bpf_maps, etc.) and a bpf program called for each kernel data object (e.g., one task, one bpf_map, etc.). User space can read kernel iterator output through *read()* syscall.

The *pin* command creates a bpf iterator from *OBJ*, and pin it to *PATH*. The *PATH* should be located in *bpffs* mount. It must not contain a dot character ('.'), which is reserved for future extensions of *bpffs*.

Map element bpf iterator requires an additional parameter *MAP* so bpf program can iterate over map elements for that map. User can have a bpf program in kernel to run with each map element, do checking, filtering, aggregation, etc. without copying data to user space.

User can then *cat PATH* to see the bpf iterator output.

bpftool iter help

Print short help message.

OPTIONS

-h, --help

Print short help message (similar to **bpftool help**).

-V, --version

Print bpftool's version number (similar to **bpftool version**), the number of the libbpf version in use, and optional features that were included when bpftool was compiled. Optional features include linking against LLVM or libbfd to provide the disassembler for JIT-*ted* programs (**bpftool prog dump jited**) and usage of BPF skeletons (some features like **bpftool prog profile** or showing pids associated to BPF objects may rely on it).

-j, --json

Generate JSON output. For commands that cannot produce JSON, this option has no effect.

-p, --pretty

Generate human-readable JSON output. Implies **-j**.

-d, --debug

Print all logs available, even debug-level information. This includes logs from libbpf as well as from the verifier, when attempting to load programs.

EXAMPLES

```
# bpftool iter pin bpf_iter_netlink.o /sys/fs/bpf/my_netlink
```

Create a file-based bpf iterator from bpf_iter_netlink.o and pin it to /sys/fs/bpf/my_netlink

```
# bpftool iter pin bpf_iter_hashmap.o /sys/fs/bpf/my_hashmap map id 20
```

Create a file-based bpf iterator from bpf_iter_hashmap.o and map with id 20, and pin it to /sys/fs/bpf/my_hashmap

SEE ALSO

bpf(2), **bpf-helpers(7)**, **bpftool(8)**, **bpftool-btf(8)**, **bpftool-cgroup(8)**, **bpftool-feature(8)**, **bpftool-gen(8)**, **bpftool-link(8)**, **bpftool-map(8)**, **bpftool-net(8)**, **bpftool-perf(8)**, **bpftool-prog(8)**, **bpftool-struct_ops(8)**, **bpftool-token(8)**