

Name

bpftool-struct_ops – tool to register/unregister/introspect BPF struct_ops

SYNOPSIS

bpftool [*OPTIONS*] **struct_ops** *COMMAND*

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

COMMANDS := { **show** | **list** | **dump** | **register** | **unregister** | **help** }

STRUCT_OPS COMMANDS

bpftool struct_ops { show | list } [STRUCT_OPS_MAP]

bpftool struct_ops dump [STRUCT_OPS_MAP]

bpftool struct_ops register OBJ [LINK_DIR]

bpftool struct_ops unregister STRUCT_OPS_MAP

bpftool struct_ops help

STRUCT_OPS_MAP := { **id** *STRUCT_OPS_MAP_ID* | **name** *STRUCT_OPS_MAP_NAME* }

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

DESCRIPTION

bpftool struct_ops { show | list } [STRUCT_OPS_MAP]

Show brief information about the struct_ops in the system. If *STRUCT_OPS_MAP* is specified, it shows information only for the given struct_ops. Otherwise, it lists all struct_ops currently existing in the system.

Output will start with struct_ops map ID, followed by its map name and its struct_ops's kernel type.

bpftool struct_ops dump [STRUCT_OPS_MAP]

Dump details information about the struct_ops in the system. If *STRUCT_OPS_MAP* is specified, it dumps information only for the given struct_ops. Otherwise, it dumps all struct_ops currently existing in the system.

bpftool struct_ops register OBJ [LINK_DIR]

Register bpf struct_ops from *OBJ*. All struct_ops under the ELF section ".struct_ops" and ".struct_ops.link" will be registered to its kernel subsystem. For each struct_ops in the ".struct_ops.link" section, a link will be created. You can give *LINK_DIR* to provide a directory path where these links will be pinned with the same name as their corresponding map name.

bpftool struct_ops unregister STRUCT_OPS_MAP

Unregister the *STRUCT_OPS_MAP* from the kernel subsystem.

bpftool struct_ops 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 struct_ops show

```
100: dctcp          tcp_congestion_ops
105: cubic          tcp_congestion_ops
```

bpftool struct_ops unregister id 105

```
Unregistered tcp_congestion_ops cubic id 105
```

bpftool struct_ops register bpf_cubic.o

```
Registered tcp_congestion_ops cubic id 110
```

SEE ALSO

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