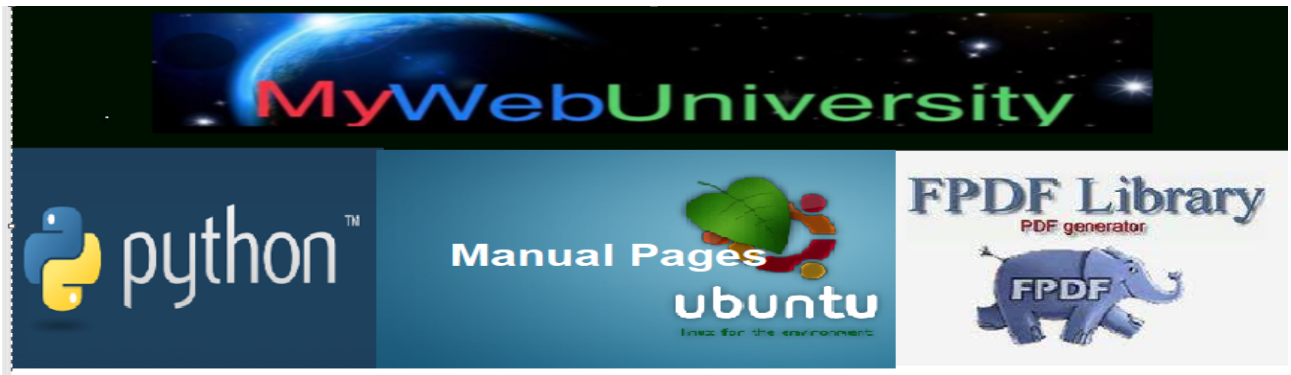


***Full credit is given to the above companies including the OS that this PDF file was generated!***

## *Linux Ubuntu 26.04 Manual Pages on command 'trace-cmd-record.1'*

***$ man trace-cmd-record.1***

TRACE-CMD-RECORD(1) libtracefs Manual TRACE-CMD-RECORD(1)

NAME

trace-cmd-record - record a trace from the Ftrace Linux internal tracer

SYNOPSIS

trace-cmd record [OPTIONS] [command]

DESCRIPTION

The trace-cmd(1) record command will set up the Ftrace Linux kernel tracer to record the specified plugins or events that happen while the command executes. If no command is given, then it will record until the user hits Ctrl-C.

The record command of trace-cmd will set up the Ftrace tracer to start tracing the various events or plugins that are given on the command line. It will then create a number of tracing processes (one per CPU) that will start recording from the kernel ring buffer straight into temporary files. When the command is complete (or Ctrl-C is hit) all the files will be combined into a trace.dat file that can later be read (see trace-cmd-report(1)).

OPTIONS

-p tracer

Specify a tracer. Tracers usually do more than just trace an event. Common tracers are: function, function_graph, preemptirqsoff, irqsoff, preemptoff and wakeup. A tracer must be supported by the running kernel. To see a list of available tracers, see trace-cmd-list(1).

-e event

Specify an event to trace. Various static trace points have been added to the Linux kernel. They are grouped by subsystem where you can enable all events of a given

subsystem or specify specific events to be enabled. The event is of the format "subsystem:event-name". You can also just specify the subsystem without the :event-name or the event-name without the "subsystem:". Using "-e sched_switch" will enable the "sched_switch" event where as, "-e sched" will enable all events under the "sched" subsystem.

The 'event' can also contain glob expressions. That is, "*stat*" will select all events (or subsystems) that have the characters "stat" in their names.

The keyword 'all' can be used to enable all events.

-a

Every event that is being recorded has its output format file saved in the output file to be able to display it later. But if other events are enabled in the trace without trace-cmd?s knowledge, the formats of those events will not be recorded and trace-cmd report will not be able to display them. If this is the case, then specify the -a option and the format for all events in the system will be saved.

-T

Enable a stacktrace on each event. For example:

```
          <idle>-0     [003] 58549.289091: sched_switch:         kworker/0:1:0 [120] R ==> trace-cmd:2603 [120]
          <idle>-0     [003] 58549.289092: kernel_stack:         <stack trace>
    => schedule (ffffffff814b260e)
    => cpu_idle (ffffffff8100a38c)
    => start_secondary (ffffffff814ab828)
```

--func-stack

Enable a stack trace on all functions. Note this is only applicable for the "function" plugin tracer, and will only take effect if the -l option is used and succeeds in limiting functions. If the function tracer is not filtered, and the stack trace is enabled, you can live lock the machine.

-f filter

Specify a filter for the previous event. This must come after a -e. This will filter what events get recorded based on the content of the event. Filtering is passed to the kernel directly so what filtering is allowed may depend on what version of the kernel you have. Basically, it will let you use C notation to check if an event should be processed or not.

==, >=, <=, >, <, &, |, && and ||

The above are usually safe to use to compare fields.

--no-filter

Do not filter out the trace-cmd threads. By default, the threads are filtered out to not be traced by events. This option will have the trace-cmd threads also be traced.

-R trigger

Specify a trigger for the previous event. This must come after a -e. This will add a given trigger to the given event. To only enable the trigger and not the event itself, then place the event after the -v option.

See Documentation/trace/events.txt in the Linux kernel source for more information on triggers.

-v

This will cause all events specified after it on the command line to not be traced. This is useful for selecting a subsystem to be traced but to leave out various events. For Example: "-e sched -v -e "*stat*"" will enable all events in the sched subsystem except those that have "stat" in their names.

Note: the *-v* option was taken from the way grep(1) inverts the following matches.

-F

This will filter only the executable that is given on the command line. If no command is given, then it will filter itself (pretty pointless). Using -F will let you trace only events that are caused by the given command.

-P pid

Similar to -F but lets you specify a process ID to trace.

-c

Used with either -F (or -P if kernel supports it) to trace the process' children too.

--user

Execute the specified command as given user.

-C clock

Set the trace clock to "clock".

Use trace-cmd(1) list -C to see what clocks are available.

-o output-file

By default, trace-cmd report will create a trace.dat file. You can specify a different

file to write to with the -o option.

-l function-name

This will limit the function and function_graph tracers to only trace the given function name. More than one -l may be specified on the command line to trace more than one function. This supports both full regex(3) parsing, or basic glob parsing. If the filter has only alphanumeric, _, *, ? and . characters, then it will be parsed as a basic glob. to force it to be a regex, prefix the filter with ^ or append it with $ and it will then be parsed as a regex.

-g function-name

This option is for the function_graph plugin. It will graph the given function. That is, it will only trace the function and all functions that it calls. You can have more than one -g on the command line.

-n function-name

This has the opposite effect of -l. The function given with the -n option will not be traced. This takes precedence, that is, if you include the same function for both -n and -l, it will not be traced.

-d

Some tracer plugins enable the function tracer by default. Like the latency tracers. This option prevents the function tracer from being enabled at start up.

-D

The option -d will try to use the function-trace option to disable the function tracer (if available), otherwise it defaults to the proc file: /proc/sys/kernel/ftrace_enabled, but will not touch it if the function-trace option is available. The -D option will disable both the ftrace_enabled proc file as well as the function-trace option if it exists.

Note, this disable function tracing for all users, which includes users outside of ftrace tracers (stack_tracer, perf, etc).

-O option

Ftrace has various options that can be enabled or disabled. This allows you to set them. Appending the text no to an option disables it. For example: "-O nograph-time" will disable the "graph-time" Ftrace option.

-s interval

The processes that trace-cmd creates to record from the ring buffer need to wake up to

do the recording. Setting the interval to zero will cause the processes to wakeup every time new data is written into the buffer. But since Ftrace is recording kernel activity, the act of this processes going back to sleep may cause new events into the ring buffer which will wake the process back up. This will needlessly add extra data into the ring buffer.

The 'interval' metric is microseconds. The default is set to 1000 (1 ms).

This is the time each recording process will sleep before waking up to record any new data that was written to the ring buffer.

-r priority

The priority to run the capture threads at. In a busy system the trace capturing threads may be staved and events can be lost. This increases the priority of those threads to the real time (FIFO) priority. But use this option with care, it can also change the behaviour of the system being traced.

-b size

This sets the ring buffer size to size kilobytes. Because the Ftrace ring buffer is per CPU, this size is the size of each per CPU ring buffer inside the kernel. Using "-b 10000" on a machine with 4 CPUs will make Ftrace have a total buffer size of 40 Megs.

--subbuf-size

The Linux kernel tracing ring buffer is broken up into sub-buffers. These sub-buffers are typically the size of the architecture "page-size". (4096 or x86). An event can only be as big as the data portion of a sub-buffer, but in most cases that?s not an issue. But the time the writer takes to switch from one sub-buffer to the next has a bit more overhead than adding events within the sub-buffer. By increasing its size, it will allow bigger events (although that is seldom an issue) but also speed up the tracing itself.

The downside of larger sub-buffers is that a "read" of the ring buffer will pull the sub-buffer size out of the ring buffer and replace it with a new sub-buffer. This may not have any real impact, but it may change the behavior slightly. Or it may not!

-B buffer-name

If the kernel supports multiple buffers, this will add a buffer with the given name. If the buffer name already exists, that buffer is just reset and will not be deleted at the end of record execution. If the buffer is created, it will be removed at the end of execution (unless the -k is set, or start command was used).

After a buffer name is stated, all events added after that will be associated with that buffer. If no buffer is specified, or an event is specified before a buffer name, it will be associated with the main (toplevel) buffer.

trace-cmd record -e sched -B block -e block -B time -e timer sleep 1

The above is will enable all sched events in the main buffer. It will then create a 'block' buffer instance and enable all block events within that buffer. A 'time' buffer instance is created and all timer events will be enabled for that event.

-m size

The max size in kilobytes that a per cpu buffer should be. Note, due to rounding to page size, the number may not be totally correct. Also, this is performed by switching between two buffers that are half the given size thus the output may not be of the given size even if much more was written.

Use this to prevent running out of diskspace for long runs.

-M cpumask

Set the cpumask for to trace. It only affects the last buffer instance given. If supplied before any buffer instance, then it affects the main buffer. The value supplied must be a hex number.

trace-cmd record -p function -M c -B events13 -e all -M 5

If the -M is left out, then the mask stays the same. To enable all CPUs, pass in a value of '-1'.

-k

By default, when trace-cmd is finished tracing, it will reset the buffers and disable all the tracing that it enabled. This option keeps trace-cmd from disabling the tracer and reseting the buffer. This option is useful for debugging trace-cmd.

Note: usually trace-cmd will set the "tracing_on" file back to what it was before it was called. This option will leave that file set to zero.

-i

By default, if an event is listed that trace-cmd does not find, it will exit with an error. This option will just ignore events that are listed on the command line but are not found on the system.

-N host:port

If another machine is running "trace-cmd listen", this option is used to have the data sent to that machine with UDP packets. Instead of writing to an output file, the data is sent off to a remote box. This is ideal for embedded machines with little storage, or having a single machine that will keep all the data in a single repository.

Note: This option is not supported with latency tracer plugins:

wakeup, wakeup_rt, irqsoff, preemptoff and preemptirqsoff

-V cid:port

If recording on a guest VM and the host is running trace-cmd listen with the -V option as well, or if this is recording on the host, and a guest in running trace-cmd listen with the -V option, then connect to the listener (the same as connecting with the -N option via the network). This has the same limitations as the -N option above with respect to latency tracer plugins.

-t

This option is used with -N, when there?s a need to send the live data with TCP packets instead of UDP. Although TCP is not nearly as fast as sending the UDP packets, but it may be needed if the network is not that reliable, the amount of data is not that intensive, and a guarantee is needed that all traced information is transfered successfully.

-q | --quiet

For use with recording an application. Suppresses normal output (except for errors) to allow only the application?s output to be displayed.

--date

With the --date option, "trace-cmd" will write timestamps into the trace buffer after it has finished recording. It will then map the timestamp to gettimeofday which will allow wall time output from the timestamps reading the created trace.dat file.

--max-graph-depth depth

Set the maximum depth the function_graph tracer will trace into a function. A value of one will only show where userspace enters the kernel but not any functions called in the kernel. The default is zero, which means no limit.

--cmdlines-size size

Set the number of entries the kernel tracing file "saved_cmdlines" can contain. This file is a circular buffer which stores the mapping between cmdlines and PIDs. If full, it leads to unresolved cmdlines ("<...>") within the trace. The kernel default value is

128.

--module module

Filter a module?s name in function tracing. It is equivalent to adding :mod:module after all other functions being filtered. If no other function filter is listed, then all modules functions will be filtered in the filter.

'--module snd' is equivalent to '-l :mod:snd'

'--module snd -l "*jack*"' is equivalent to '-l "*jack*:mod:snd"'

'--module snd -n "*"' is equivalent to '-n :mod:snd'

--proc-map

Save the traced process address map into the trace.dat file. The traced processes can be specified using the option -P, or as a given command.

--profile

With the --profile option, "trace-cmd" will enable tracing that can be used with trace-cmd-report(1) --profile option. If a tracer -p is not set, and function graph depth is supported by the kernel, then the function_graph tracer will be enabled with a depth of one (only show where userspace enters into the kernel). It will also enable various tracepoints with stack tracing such that the report can show where tasks have been blocked for the longest time.

See trace-cmd-profile(1) for more details and examples.

-G

Set interrupt (soft and hard) events as global (associated to CPU instead of tasks). Only works for --profile.

-H event-hooks

Add custom event matching to connect any two events together. When not used with --profile, it will save the parameter and this will be used by trace-cmd report --profile, too. That is:

trace-cmd record -H hrtimer_expire_entry,hrtimer/hrtimer_expire_exit,hrtimer,sp

trace-cmd report --profile

Will profile hrtimer_expire_entry and hrtimer_expire_ext times.

See trace-cmd-profile(1) for format.

-S

(for --profile only) Only enable the tracer or events speficied on the command line. With this option, the function_graph tracer is not enabled, nor are any events (like

sched_switch), unless they are specifically specified on the command line (i.e. -p function -e sched_switch -e sched_wakeup)

--temp directory

When trace-cmd is recording the trace, it records the per CPU data into a separate file for each CPU. At the end of the trace, these files are concatenated onto the final trace.dat file. If the final file is on a network file system, it may not be appropriate to copy these temp files into the same location. --temp can be used to tell trace-cmd where those temp files should be created.

--ts-offset offset

Add an offset for the timestamp in the trace.dat file. This will add a offset option into the trace.dat file such that a trace-cmd report will offset all the timestamps of the events by the given offset. The offset is in raw units. That is, if the event timestamps are in nanoseconds the offset will also be in nanoseconds even if the displayed units are in microseconds.

--tsync-interval

Set the loop interval, in ms, for timestamps synchronization with guests: If a negative number is specified, timestamps synchronization is disabled If 0 is specified, no loop is performed - timestamps offset is calculated only twice, at the beginning and at the end of the trace. Timestamps synchronization with guests works only if there is support for VSOCK.

--tsc2nsec

Convert the current clock to nanoseconds, using tsc multiplier and shift from the Linux kernel?s perf interface. This option does not change the trace clock, just assumes that the tsc multiplier and shift are applicable for the selected clock. You may use the "-C tsc2nsec" clock, if not sure what clock to select.

--stderr

Have output go to stderr instead of stdout, but the output of the command executed will not be changed. This is useful if you want to monitor the output of the command being executed, but not see the output from trace-cmd.

--poll

Waiting for data to be available on the trace ring-buffers may trigger IPIs. This might generate unacceptable trace noise when tracing low latency or real time systems. The poll option forces trace-cmd to use O_NONBLOCK. Traces are extracted by busy waiting,

which will hog the CPUs, so only use when really needed.

--name

Give a specific name for the current agent being processed. Used after -A to give the guest being traced a name. Useful when using the vsocket ID instead of a name of the guest.

--verbose[=level]

Set the log level. Supported log levels are "none", "critical", "error", "warning", "info", "debug", "all" or their identifiers "0", "1", "2", "3", "4", "5", "6". Setting the log level to specific value enables all logs from that and all previous levels. The level will default to "info" if one is not specified.

Example: enable all critical, error and warning logs

trace-cmd record --verbose=warning

--file-version

Desired version of the output file. Supported versions are 6 or 7.

--compression

Compression of the trace output file, one of these strings can be passed:

'any' - auto select the best available compression algorithm

'none' - do not compress the trace file

'name' - the name of the desired compression algorithms. Available algorithms can be listed with

trace-cmd list -c

--proxy vsocket

Use a vsocket proxy to reach the agent. Acts the same as -A (for an agent) but will send the proxy connection to the agent. It is expected to run on a privileged guest that the host is aware of (as denoted by the cid in the -P option for the agent).

--daemonize Run trace-cmd in the background as a daemon after recording has started. Creates a pidfile at /var/run/trace-cmd-record.pid with the pid of trace-cmd during the recording.

EXAMPLES

The basic way to trace all events:

```
# trace-cmd record -e all ls > /dev/null
# trace-cmd report
    trace-cmd-13541 [003] 106260.693809: filemap_fault: address=0x128122 offset=0xce
        trace-cmd-13543 [001] 106260.693809: kmalloc: call_site=81128dd4 ptr=0xffff88003dd83800 bytes_req=768
bytes_alloc=1024 gfp_flags=GFP_KERNEL|GFP_ZERO
```

```
            ls-13545 [002] 106260.693809: kfree: call_site=810a7abb ptr=0x0
            ls-13545 [002] 106260.693818: sys_exit_write:       0x1
```

To use the function tracer with sched switch tracing:

```
    # trace-cmd record -p function -e sched_switch ls > /dev/null
    # trace-cmd report
            ls-13587 [002] 106467.860310: function: hrtick_start_fair <-- pick_next_task_fair
                ls-13587 [002] 106467.860313: sched_switch: prev_comm=trace-cmd prev_pid=13587 prev_prio=120
prev_state=R ==> next_comm=trace-cmd next_pid=13583 next_prio=120
          trace-cmd-13585 [001] 106467.860314: function: native_set_pte_at <-- __do_fault
          trace-cmd-13586 [003] 106467.860314: function:             up_read <-- do_page_fault
            ls-13587 [002] 106467.860317: function:             __phys_addr <-- schedule
          trace-cmd-13585 [001] 106467.860318: function: _raw_spin_unlock <-- __do_fault
            ls-13587 [002] 106467.860320: function: native_load_sp0 <-- __switch_to
          trace-cmd-13586 [003] 106467.860322: function: down_read_trylock <-- do_page_fault
```

Here is a nice way to find what interrupts have the highest latency:

```
    # trace-cmd record -p function_graph -e irq_handler_entry  -l do_IRQ sleep 10
    # trace-cmd report
      <idle>-0     [000] 157412.933969: funcgraph_entry:                |  do_IRQ() {
      <idle>-0     [000] 157412.933974: irq_handler_entry:    irq=48 name=eth0
      <idle>-0     [000] 157412.934004: funcgraph_exit:       + 36.358 us |  }
      <idle>-0     [000] 157413.895004: funcgraph_entry:                |  do_IRQ() {
      <idle>-0     [000] 157413.895011: irq_handler_entry:    irq=48 name=eth0
      <idle>-0     [000] 157413.895026: funcgraph_exit:                 + 24.014 us |  }
      <idle>-0     [000] 157415.891762: funcgraph_entry:                |  do_IRQ() {
      <idle>-0     [000] 157415.891769: irq_handler_entry:    irq=48 name=eth0
      <idle>-0     [000] 157415.891784: funcgraph_exit:       + 22.928 us |  }
      <idle>-0     [000] 157415.934869: funcgraph_entry:                |  do_IRQ() {
      <idle>-0     [000] 157415.934874: irq_handler_entry:    irq=48 name=eth0
      <idle>-0     [000] 157415.934906: funcgraph_exit:       + 37.512 us |  }
      <idle>-0     [000] 157417.888373: funcgraph_entry:                |  do_IRQ() {
      <idle>-0     [000] 157417.888381: irq_handler_entry:    irq=48 name=eth0
      <idle>-0     [000] 157417.888398: funcgraph_exit:       + 25.943 us |  }
```

An example of the profile:

```
 # trace-cmd record --profile sleep 1
 # trace-cmd report --profile --comm sleep
task: sleep-21611
  Event: sched_switch:R (1) Total: 99442 Avg: 99442 Max: 99442 Min:99442
    <stack> 1 total:99442 min:99442 max:99442 avg=99442
      => ftrace_raw_event_sched_switch (0xffffffff8105f812)
      => __schedule (0xffffffff8150810a)
      => preempt_schedule (0xffffffff8150842e)
      => ___preempt_schedule (0xffffffff81273354)
      => cpu_stop_queue_work (0xffffffff810b03c5)
      => stop_one_cpu (0xffffffff810b063b)
      => sched_exec (0xffffffff8106136d)
      => do_execve_common.isra.27 (0xffffffff81148c89)
      => do_execve (0xffffffff811490b0)
      => SyS_execve (0xffffffff811492c4)
      => return_to_handler (0xffffffff8150e3c8)
      => stub_execve (0xffffffff8150c699)
  Event: sched_switch:S (1) Total: 1000506680 Avg: 1000506680 Max: 1000506680 Min:1000506680
    <stack> 1 total:1000506680 min:1000506680 max:1000506680 avg=1000506680
      => ftrace_raw_event_sched_switch (0xffffffff8105f812)
      => __schedule (0xffffffff8150810a)
      => schedule (0xffffffff815084b8)
      => do_nanosleep (0xffffffff8150b22c)
      => hrtimer_nanosleep (0xffffffff8108d647)
      => SyS_nanosleep (0xffffffff8108d72c)
      => return_to_handler (0xffffffff8150e3c8)
      => tracesys_phase2 (0xffffffff8150c304)
  Event: sched_wakeup:21611 (1) Total: 30326 Avg: 30326 Max: 30326 Min:30326
    <stack> 1 total:30326 min:30326 max:30326 avg=30326
      => ftrace_raw_event_sched_wakeup_template (0xffffffff8105f653)
      => ttwu_do_wakeup (0xffffffff810606eb)
      => ttwu_do_activate.constprop.124 (0xffffffff810607c8)
      => try_to_wake_up (0xffffffff8106340a)
```

An example of using --daemonize together with guest/host tracing:

```
$ sudo trace-cmd record --daemonize -p nop -e 'sched:sched_process_exec' -A guest -p nop -e net &&
> ping -c 1 10.20.1.2 &&
> sudo start-stop-daemon --stop --signal INT --retry 20 --pidfile /var/run/trace-cmd-record.pid &&
> sudo trace-cmd report -i trace.dat -i trace-guest.dat | head
Negotiated kvm time sync protocol with guest guest
Send SIGINT to pid 3071371 to stop recording
PING 10.20.1.2 (10.20.1.2) 56(84) bytes of data.
64 bytes from 10.20.1.2: icmp_seq=1 ttl=64 time=0.134 ms
--- 10.20.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.134/0.134/0.134/0.000 ms
CPU0 data recorded at offset=0x14f000
    229 bytes in size (4096 uncompressed)
....
      trace.dat: cpus=28
trace-guest.dat: cpus=1
          trace.dat:          ping-3071450 [013] 1196830.834258: sched_process_exec:     filename=/bin/ping pid=3071450
old_pid=3071450
          trace-guest.dat:          <idle>-0       [000] 1196830.835990: napi_gro_receive_entry: dev=eth1 napi_id=0x2002
queue_mapping=1  skbaddr=0xffff95d051a5c400  vlan_tagged=0  vlan_proto=0x0000  vlan_tci=0x0000  protocol=0x0800
ip_summed=0  hash=0x00000000  l4_hash=0  len=84  data_len=0  truesize=768  mac_header_valid=1  mac_header=-14
nr_frags=0 gso_size=0 gso_type=0
trace-guest.dat:          <idle>-0     [000] 1196830.835997: napi_gro_receive_exit:  ret=3
          trace-guest.dat:          <idle>-0          [000]  1196830.835998:  netif_receive_skb:          dev=eth1
skbaddr=0xffff95d051a5c400x len=84
          trace-guest.dat:          <idle>-0          [000]  1196830.836021:  net_dev_queue:          dev=eth1
skbaddr=0xffff95d051a5c700x len=98
          trace-guest.dat:          <idle>-0       [000] 1196830.836024: net_dev_start_xmit:      dev=eth1 queue_mapping=0
skbaddr=0xffff95d051a5c700  vlan_tagged=0  vlan_proto=0x0000  vlan_tci=0x0000  protocol=0x0800  ip_summed=0  len=98
data_len=0 network_offset=14 transport_offset_valid=1 transport_offset=34 tx_flags=0 gso_size=0 gso_segs=0 gso_type=0
          trace-guest.dat:          <idle>-0          [000]  1196830.836069:  net_dev_xmit:          dev=eth1
skbaddr=0xffff95d051a5c700 len=98 rc=0
```

trace.dat: sudo-3071451 [015] 1196830.838262: sched_process_exec: filename=/usr/bin/sudo pid=3071451 old_pid=3071451

SEE ALSO

trace-cmd(1), trace-cmd-report(1), trace-cmd-start(1), trace-cmd-stop(1), trace-cmd-extract(1), trace-cmd-reset(1), trace-cmd-split(1), trace-cmd-list(1), trace-cmd-listen(1), trace-cmd-profile(1)

AUTHOR

Written by Steven Rostedt, <rostedt@goodmis.org[1]>

RESOURCES

https://git.kernel.org/pub/scm/utils/trace-cmd/trace-cmd.git/

COPYING



NOTES

1. rostedt@goodmis.org

   mailto:rostedt@goodmis.org