

ftrace
dem Kernel bei der Arbeit zusehen

FrOSCon
15. August 2026

Martin Kaiser

Goals of this talk

- hands-on introduction to ftrace
 - examples not only for developers
- encourage you to try ftrace
- make sure you remember ftrace next time you debug a problem

About me

- Codasip (www.codasip.com)
 - Linux Kernel on RISC-V
 - CHERI (www.cheri-linux.org)
- past
 - Linux Kernel on ARM (i.MX)
 - RFID (ISO 14443)
 - Wireshark
- www.kaiser.cx

ftrace

- tracing framework for the Linux Kernel
- started in 2008, not frequently used (I think)
- function calls
- events
- ringbuffer

design goals

- minimum overhead
 - kernel with ftrace support, tracing disabled
 - distro kernels support ftrace
- require no additional tools
 - only echo, cat

tracefs

- pseudo filesystem, similar to sysfs
- `/sys/kernel/tracing/`

<code>current_tracer</code>	select a tracer
<code>tracing_on</code>	enable/disable tracing
<code>trace</code> , <code>trace_pipe</code>	the ringbuffer
<code>README</code>	documentation

- files to configure filters, options
 - options modify the behaviour of tracers

function tracer

- intercept and trace function calls
- which functions can be traced?
 - */sys/kernel/tracing/available_filter_functions*
- functions that can't be traced
 - inline functions
 - functions that implement tracing
 - functions that are called very early at startup

function tracer (II)

```
[root@nb282 ~]# cd /sys/kernel/tracing/
[root@nb282 tracing]# echo function > current_tracer
[root@nb282 tracing]# echo 1 > tracing_on
[root@nb282 tracing]# echo 0 > tracing_on
[root@nb282 tracing]# head -12 trace
# tracer: function
#
#          _-----=> irqsoff
#          / _-----=> need-resched
#          | / _---=> hardirq/softirq
#          || / _--=> preempt-depth
#          ||| /      delay
#          TASK-PID   CPU#  | TIMESTAMP | FUNCTION
#          | |       |   |   |   |   |
<idle>-0    [016] d... 312279.375138: rcu_idle_exit <-cpuidle_enter_state
<idle>-0    [018] d... 312279.375139: rcu_idle_exit <-cpuidle_enter_state
<idle>-0    [019] d... 312279.375139: rcu_idle_exit <-cpuidle_enter_state
[root@nb282 tracing]#
```

filters

- simple syntax
 - it's all implemented in the kernel
- function names, wildcards (* and ?)
 - exclude functions
- PID, children
 - function-fork option

filters

```
[root@nb282 ~]# cd /sys/kernel/tracing/
[root@nb282 tracing]# echo "vfs*" > set_ftrace_filter
[root@nb282 tracing]# echo "*read*" > set_ftrace_notrace
[root@nb282 tracing]# echo function > current_tracer
[root@nb282 tracing]# echo 1 > tracing_on
[root@nb282 tracing]# echo 0 > tracing_on
[root@nb282 tracing]# head -12 trace
# tracer: function
#
#          _-----=> irqsoff
#          / _-----=> need-resched
#          | / _---=> hardirq/softirq
#          || / _---=> preempt-depth
#          ||| /      delay
#          # TASK-PID    CPU#    ||||    TIMESTAMP    FUNCTION
#          #   | |       |   ||||    |             |
qemu-system...-1300780 [004] .... 312895.914814: vfs_write <-ksys_write
                bash-18279 [015] .... 312895.914863: vfs_write <-ksys_write
                bash-18279 [015] .... 312895.915000: vfs_write <-ksys_write
[root@nb282 tracing]#
```

demo

- terminal emulator (rxvt), shell (bash)
- press a key
 - terminal receives the key
 - terminal sends a character to the shell
 - character device
 - line discipline
 - hardware interface (pseudo tty)
- use ftrace to see what happens under the hood

demo
function tracer, filtering

trace-cmd

- cmdline frontend for ftrace
 - uses tracefs internally
- packaged everywhere
- most important command: trace-cmd reset
- trace-cmd start, stop, show
- trace-cmd record, report
 - kernelshark GUI

trace-cmd (II)

```
[root@nb282 ~]# trace-cmd start -p function -l "vfs*" -n "*read*"
  plugin 'function'
[root@nb282 ~]# trace-cmd stop
[root@nb282 ~]# trace-cmd show | head -12
# tracer: function
#
#
#          _-----=> irqs-off
#         / _-----=> need-resched
#        | / _-----=> hardirq/softirq
#       || / _-----=> preempt-depth
#      ||| /
#     ||| /      delay
#
#          TASK-PID      CPU#      ||||      TIMESTAMP      FUNCTION
#          | |          |         ||||      |              |
bash-5383 [004]   ....   313558.601214:  vfs_write  <-ksys_write
bash-5383 [004]   ....   313558.601216:  vfs_statx  <-__do_sys_newstat
rxvt-5077 [012]   ....   313558.601214:  vfs_write  <-ksys_write
[root@nb282 ~]#
```

function graph tracer

- intercept start and end of each traced function
- graph structure of function calls
- track the time spent in a function

function graph tracer (II)

```
[root@nb282 ~]# trace-cmd start -p function_graph
  plugin 'function_graph'
[root@nb282 ~]# trace-cmd stop ; trace-cmd show
...
18)          |          hrtimer_update_next_event() {
18)          |          __hrtimer_get_next_event() {
18)    0.024 us |          __hrtimer_next_event_base();
18)    0.249 us |          }
18)          |          __hrtimer_get_next_event() {
          |          ...
18)    0.276 us |          }
18)    0.931 us |          }
[root@nb282 ~]#
```

demo
trace-cmd, function graph

events

- predefined points that can be traced
 - linked to a set of data items
- */sys/kernel/tracing/events/<group>/<event>/*

```
[root@nb282 ~]# cd /sys/kernel/tracing/  
[root@nb282 tracing]# ls events/workqueue/  
... workqueue_activate_work/    workqueue_queue_work/  
... workqueue_execute_start/    workqueue_execute_end/  
[root@nb282 tracing]# ls events/workqueue/workqueue_queue_work/  
enable filter format hist id trigger  
[root@nb282 tracing]#
```

events (II)

```
[root@nb282 ~]# cd /sys/kernel/tracing/
[root@nb282 tracing]# cat events/workqueue/workqueue_queue_work/format
name: workqueue_queue_work
ID: 266
format:
    field:unsigned short common_type;          offset:0; size:2; signed:0;
    field:unsigned char common_flags;          offset:2; size:1; signed:0;
    field:unsigned char common_preempt_count;  offset:3; size:1; signed:0;
    field:int common_pid;                      offset:4; size:4; signed:1;

    field:void * work;                          offset:8; size:8; signed:0;
    field:void * function;                      offset:16; size:8; signed:0;
    field:void * workqueue;                    offset:24; size:8; signed:0;
    field:unsigned int req_cpu;                offset:32; size:4; signed:0;
    field:unsigned int cpu;                    offset:36; size:4; signed:0;

print fmt: "work struct=%p function=%ps workqueue=%p req_cpu=%u cpu=%u",
          REC->work, REC->function, REC->workqueue, REC->req_cpu, REC->cpu
[root@nb282 tracing]#
```

events (III)

- the data fields can be used for filtering
- trigger : action when a filter matches
 - e.g. enable function tracing, show a stacktrace

events (IV)

```
[root@nb282 ~]# trace-cmd list -e mmc
mmc:mmc_request_start
mmc:mmc_request_done
[root@nb282 ~]# trace-cmd list -e mmc:mmc_request_done -F
system: mmc
name: mmc_request_done
ID: 2847
format:
    ...
    field:int cmd_err; offset:12;  size:4;  signed:1;
    ...
[root@nb282 ~]# trace-cmd start -e mmc:mmc_request_done -f "cmd_err != 0"
[root@nb282 ~]#
```

event demo

probes

- a dynamic event
- configurable at runtime
 - what triggers the dynamic event?
 - what fields does the event have?
- */sys/kernel/tracing/dynamic_events*
- supported by distro kernels

probe types

- probe type defines the type of trigger for the dynamic event
 - fprobe : function entry / exit
 - kprobe : (almost) any kernel instruction
 - ...
- available fields for fprobe, kprobe
 - function arguments, return value
 - stack contents
 - arbitrary memory (e.g. value of a global symbol)

fprobe example

```
static int n_tty_open(struct tty_struct *tty);
```

```
# echo 'f:demo1 n_tty_open result=$retval:s8 tty_param=$arg1' > ./dynamic_events
# ls events/fprobes/demo1/
enable  filter  format  id  inject  trigger
# cat events/fprobes/demo1/format
...
field:s8  result;      offset:24; size:1; signed:1;
field:u64 tty_param;  offset:25; size:8; signed:0;
...

# trace-cmd start -e fprobes
# cat /dev/ttyS3
cat: /dev/ttyS3: Input/output error
# trace-cmd show | tail -1
... demo1: (tty_ldisc_setup+0x30/0xb0 <- n_tty_open) result=0 tty_param=0xffff88800722b400
#
```

fprobe demo

bug analysis with ftrace

```
[martin.kaiser@nb282 ~]$ grep test /*/*/*  
bash: /usr/bin/grep: Argument list too long  
[martin.kaiser@nb282 ~]$
```

- who is at fault?
 - grep? bash? libc? kernel?
- what's going wrong?

bug analysis demo

ftrace implementation

```
[root@nb282 tracing]# trace-cmd start -p function ...  
[root@nb282 tracing]# trace-cmd show  
bash-5383 [004] .... 313558.601214: vfs_write <-ksys_write
```

intercept function calls

```
ssize_t vfs_write(struct file *file, ...)  
{  
    if (tracing_enabled_for_this_function)  
        write_log_entry();  
  
    ... do our work  
}
```

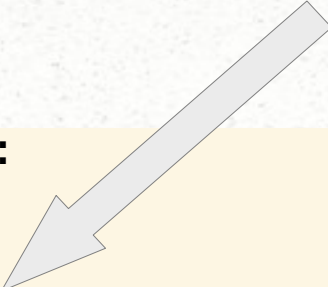
- additional check in each function that can be traced
 - condition is almost always false (tracing is off)
- > this approach is too slow

ftrace implementation (II)

- on/off switch: “tracing on for this function?”
 - checked very often -> must be fast
 - hardly ever switched -> can be slow
- patch the code at runtime
 - depends on compiler, architecture

ftrace implementation (III)

- make the compiler generate “empty code” at the start
 - nop == no operation



```
ffffffff803a0960 <vfs_write>:  
ffffffff803a0960: 0001  nop  
ffffffff803a0962: 0001  nop  
ffffffff803a0964: 0001  nop  
ffffffff803a0966: 0001  nop  
ffffffff803a0968: 714d  caddi    csp, csp, -0x150  
ffffffff803a096a: a286  sc      cra, 0x140(csp)
```

- tracing off : keep the 4 x nop
- tracing on : replace 4 x nop with jump to log function

Thank you for your attention.

Questions?

Slides at www.kaiser.cx



*cmdline sessions
of the demos*

demo: function tracer, filtering

```
[root@nb282 tracing]# echo nop > current_tracer
[root@nb282 tracing]# echo "" > set_ftrace_filter
[root@nb282 tracing]# echo "" > set_ftrace_pid
[root@nb282 tracing]# echo 0 > tracing_on
[root@nb282 tracing]# echo > trace
```

```
[martin.kaiser@nb282 ~]$ echo $$
115763
[martin.kaiser@nb282 ~]$ pstree -s -p $$
systemd(1)──...──rxvt(115762)──bash(115763)──pstree(115834)
```

```
[root@nb282 tracing]# echo "*write*" > set_ftrace_filter
[root@nb282 tracing]# echo 115762 > set_ftrace_pid
[root@nb282 tracing]# echo function > current_tracer
[root@nb282 tracing]# echo 1 > tracing_on
[root@nb282 tracing]# cat trace_pipe
```

```
[martin.kaiser@nb282 ~]$ e
```

demo: function tracer, filtering (II)

```
rxvt-115762 [008] .... 73198.438343: __x64_sys_write <-do_syscall_64
rxvt-115762 [008] .... 73198.438361: ksys_write <-do_syscall_64
rxvt-115762 [008] .... 73198.438362: vfs_write <-ksys_write
rxvt-115762 [008] .... 73198.438382: __vfs_write <-vfs_write
rxvt-115762 [008] .... 73198.438383: tty_write <-vfs_write
rxvt-115762 [008] .... 73198.438386: tty_write_lock <-tty_write
rxvt-115762 [008] .... 73198.438388: n_tty_write <-tty_write
rxvt-115762 [008] .... 73198.438391: pty_write <-n_tty_write
rxvt-115762 [008] .... 73198.438400: tty_write_unlock <-tty_write
rxvt-115762 [008] .... 73198.439042: tty_write_room <-n_tty_poll
rxvt-115762 [008] .... 73198.439058: pty_write_room <-n_tty_poll
rxvt-115762 [008] .... 73198.439136: tty_write_room <-n_tty_poll
rxvt-115762 [008] .... 73198.439141: pty_write_room <-n_tty_poll
```

demo: trace-cmd, function graph tracer

```
root@nb282 tracing]# trace-cmd reset ; trace-cmd start \  
-p function -P 115762 -l "*write*"  
  plugin 'function'  
[root@nb282 tracing]# cat trace_pipe
```

```
[martin.kaiser@nb282 ~]$ e
```

```
...  
rxvt-115762 [014] .... 73524.534909: pty_write_room <-n_tty_poll  
rxvt-115762 [014] .... 73524.552243: __x64_sys_writev <-do_syscall_64  
rxvt-115762 [014] .... 73524.552252: do_writev <-do_syscall_64  
rxvt-115762 [014] .... 73524.552252: vfs_writev <-do_writev
```

^C

```
[root@nb282 tracing]# trace-cmd stop  
[root@nb282 tracing]#
```

demo: trace-cmd, function graph tracer (II)

```
[root@nb282 tracing]# trace-cmd reset
[root@nb282 tracing]# trace-cmd start -p function_graph -P 115762 \
    -g __x64_sys_write --max-graph-depth 10 \
    -n smp_irq_work_interrupt -n kprobe_ftrace_handler
    plugin 'function_graph'
[root@nb282 tracing]# cat trace_pipe
```

```
[martin.kaiser@nb282 ~]$ e
```

```
8)      |  __x64_sys_write() {
...
8)      |                      pty_write() {
8)      |                      tty_insert_flip_string_and_push_buffer() {
...
8)      |                      tty_insert_flip_string_fixed_flag.part.9() {
...
8)      |                      queue_work_on() {
8) 4.876 us |                      __queue_work();
8) 5.963 us |                      }
...

```

event demo

```
[root@nb282 tracing]# trace-cmd reset
[root@nb282 tracing]# trace-cmd start -p function_graph \
-P 12223 -g pty_write --max-graph-depth 5 \
-n "*mutex*" -n "*spin_lock*" -n smp_irq_work_interrupt \
-O nofuncgraph-duration -e workqueue
plugin 'function_graph'
[root@nb282 tracing]# cat trace_pipe
```

```
[martin.kaiser@nb282 ~]$ e
```

```
8) pty_write() {
...
8)      queue_work_on() {
8)      __queue_work() {
8)          /* workqueue_queue_work: work struct=000000009b958723
           function=flush_to_ldisc workqueue=0000000098641a65
           req_cpu=8192 cpu=4294967295 */
8)          /* workqueue_activate_work: work struct 000000009b958723 */
8)      }
```

fprobe demo

```
[root@nb282 tracing]# trace-cmd reset
[root@nb282 tracing]# echo 'f:p1 n_tty_write c=+0($arg3):char num=$arg4' > \
    /sys/kernel/tracing/dynamic_events
[root@nb282 tracing]# trace-cmd start -p function_graph \
    -P 12223 -g __x64_sys_write --max-graph-depth 5 \
    -n "*mutex*" -n "*spin_lock*" -n smp_irq_work_interrupt \
    -O nofuncgraph-duration -e workqueue -e fprobes:p1
    plugin 'function_graph'
[root@nb282 tracing]# cat trace_pipe
```

```
[martin.kaiser@nb282 ~]$ d
```

```
__x64_sys_write() {
    ...
    tty_write() {
        ...
        n_tty_write() {
            /* p1: (n_tty_write+0x0/0x3f0) c='d' num=0x1 */
            ...
            pty_write() {
                ...
            }
        }
    }
}
```

bug analysis demo

This command fails

```
martin@reykholt:~$ grep test /*/*/*  
-bash: /usr/bin/grep: Argument list too long
```

See if we can use strace to get a first idea.

```
martin@reykholt:~$ strace -f grep test /*/*/*  
-bash: /usr/bin/grep: Argument list too long
```

(user shell has pid 3630)

```
root@reykholt:~# strace -f -p 3630
```

```
martin@reykholt:~$ grep test /*/*/*
```

```
[pid 3765] execve("/usr/bin/grep", ["grep", "test",  
"/bin/X11/[, ...], 0x557042ee8130 /* 28 vars */) = -1 E2BIG (Argument  
list too long)
```

bug analysis demo

or use ftrace to see failing syscalls from children

```
root@reykholt:~# trace-cmd start -P 3630 -c \  
-e raw_syscalls:sys_exit -f "common_pid != 3630 && ret < 0"  
root@reykholt:~# cat /sys/kernel/tracing/trace_pipe
```

```
martin@reykholt:~$ grep test /*/*/*
```

```
<...>-3943      [003] ..... 2757.394452: sys_exit: NR 59 = -7  
...
```

the execve syscall fails -> try to figure out what happens internally
look for functions related to execve where we could start tracing

```
root@reykholt:~# trace-cmd list -f execve  
do_execveat_common.isra.0  
kernel_execve  
__x64_sys_execve  
...
```

bug analysis demo

- watch events for entering and exiting execve* syscalls
 - enable tracing on entry, disable tracing on exit
- create a call tree for do_execveat_common.isra.0
 - filter out irrelevant functions that are called frequently

```
root@reykholt:~# trace-cmd reset
root@reykholt:~# trace-cmd start -p function_graph -P 3630 -c \
-e syscalls:sys_enter_execve* -R traceon \
-e syscalls:sys_exit_execve* -R traceoff \
-g do_execveat_common.isra.0 --max-graph-depth 3 \
-O function-fork -O funcgraph-tail -O nofuncgraph-duration \
-O nofuncgraph-cpu \
-n handle_mm_fault -n __cond_resched -n mm_alloc \
-n mmput -n vm_area_alloc -n insert_vm_struct
```

```
martin@reykholt:~$ grep test /*/*/*
```

bug analysis demo

```
root@reykholt:~# cat /sys/kernel/tracing/trace_pipe

/* sys_execve(filename: 557042f5a6f0, argv: ..., envp: ...) */
do_execveat_common.isra.0() {
    alloc_bprm() {
        do_open_execat();
        ...
    } /* alloc_bprm */
    ...
    bprm_stack_limits();
    free_bprm() {
        ...
    } /* free_bprm */
    putname() {
        kmem_cache_free();
    } /* putname */
} /* do_execveat_common.isra.0 */
/* sys_execve -> 0xffffffffffffff9 */
```

- sequence
 - alloc_...
 - bprm_stack_limits
 - free_...
 - > does bprm_stack_limits fail?

bug analysis demo

- same commands as in the first run
- add an fprobe to see the return value of bprm_stack_limits()
- enable this new probe while tracing

```
root@reykholt:~# trace-cmd reset
root@reykholt:~# echo 'f bprm_stack_limits ret=$retval' > \
    /sys/kernel/tracing/dynamic_events
root@reykholt:~# trace-cmd start -p function_graph -P 3630 -c \
    -e syscalls:sys_enter_execve* -R traceon \
    -e syscalls:sys_exit_execve* -R traceoff \
    -g do_execveat_common.isra.0 --max-graph-depth 3 \
    -O function-fork -O funcgraph-tail -O nofuncgraph-duration \
    -O nofuncgraph-cpu \
    -n handle_mm_fault -n __cond_resched -n mm_alloc \
    -n mmput -n vm_area_alloc -n insert_vm_struct \
    -e fprobes
```

```
martin@reykholt:~$ grep test /*/*/*
```

bug analysis demo

```
root@reykholt:~# cat /sys/kernel/tracing/trace_pipe

/* sys_execve(filename: 557042f5a6f0, argv: ..., envp: ...) */
do_execveat_common.isra.0() {
    alloc_bprm() {
        ...
    } /* alloc_bprm */
    ...
    bprm_stack_limits() {
        /* bprm_stack_limits__exit: (return_to_handler+0x0/0x50
        <- bprm_stack_limits) ret=-7 */
    } /* bprm_stack_limits */
    ...
    free_bprm() {
        ...
    } /* free_bprm */
    ...
} /* do_execveat_common.isra.0 */
/* sys_execve -> 0xffffffffffffffff9 */
```

bug analysis demo

- bprm_stack_limits function
 - small, mostly comments
 - "Limit to 1/4 of the max stack size ... for the argv+env strings."

```
martin@reykholt:~$ ulimit -a
...
stack size      (kbytes, -s) 512

martin@reykholt:~$ grep test /*/*/*
-bash: /usr/bin/grep: Argument list too long
```

bug analysis demo

open a new shell -> the default stack size is 8K

```
martin@reykholt:~$ ulimit -s  
8192  
martin@reykholt:~$ grep test /*/*/*  
... lots of output, no error
```

bonus slides

function graph tracer : filters

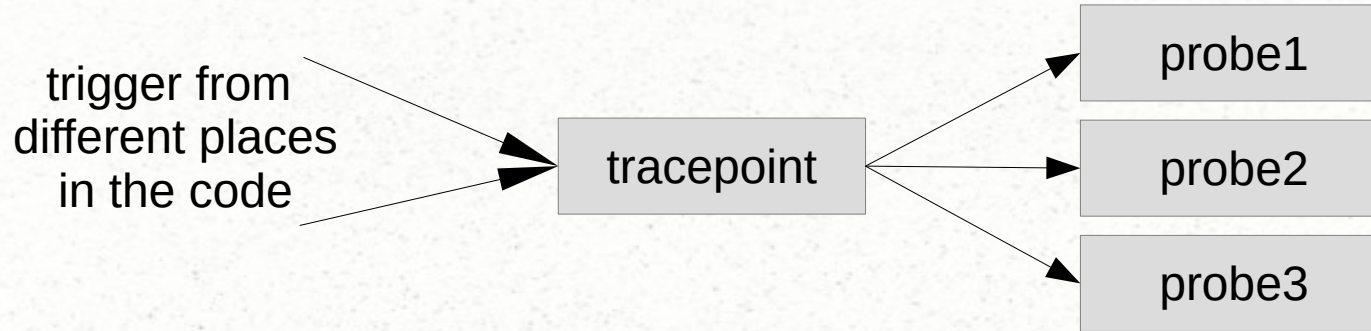
- two different filters
 - list the function?
 - start function graph?

	/sys/kernel/tracing/	trace-cmd start	action
function, enable	set_ftrace_filter	-l <func>	list the function no graph
graph, enable	set_graph_function	-g <func>	start graph tracing
disable	set_ftrace_notrace set_graph_notrace	-n <func>	disable tracing

tracepoints

- abstract concept that is the basis for events
- tracepoint == name, data fields
- probe function(s)
 - can be provided at runtime (modules, BPF)
- trigger the tracepoint -> probe(s) are called

tracepoints (II)



- event
 - tracepoint + probe that logs into the ringbuffer
- software that uses tracepoints, probes
 - e.g. Sysdig, SentinelOne, perf

perf

- one of the most prominent users of ftrace
- a performance analyzing tool
 - bundled with the linux kernel
 - actively developed
- interface to function (graph) tracers
 - uses tracefs

perf

- perf events
 - hardware performance counters
 - software events
 - ...
 - tracepoint events
 - they use the tracepoints from ftrace
 - with perf-specific probe functions

perf

```
[root@nb282 ~]# perf ftrace -G "kmem*" --graph-opts noirqs sleep 0.1
# tracer: function_graph
#
# CPU      DURATION      FUNCTION CALLS
# |         |         |           |         |         |
4)         |         |         |         |         |
4)         |         |         |         |         |
4) 0.147 us |         |         |         |         |
4) 1.051 us |         |         |         |         |
4) 0.191 us |         |         |         |         |
4) 3.613 us |         |         |         |         |
...
[root@nb282 ~]#
```

perf

```
[root@nb282 ~]# perf list
```

bus-cycles	[Hardware event]
cache-misses	[Hardware event]
...	
context-switches OR cs	[Software event]
page-faults OR faults	[Software event]
...	
mmc:mmc_request_done	[Tracepoint event]
mmc:mmc_request_start	[Tracepoint event]
...	

```
[root@nb282 ~]# █
```

perf

```
[root@nb282 ~]# perf stat -e "timer:hrtimer*" -e cs -e page-faults sleep 0.1
```

Performance counter stats for 'sleep 0.1':

1	timer:hrtimer_init
3	timer:hrtimer_start
2	timer:hrtimer_expire_entry
2	timer:hrtimer_expire_exit
2	timer:hrtimer_cancel
1	cs
57	page-faults

0.102325429 seconds time elapsed

...
[root@nb282 ~]# █