Tracing with ftrace
Critical tooling for Linux Development

Steven Rostedt

June 4, 2021

©2021 VMware, Inc.

What is ftrace?

- The official tracer for Linux (since 2008)

vmware®
©2021 VMware, Inc.

What is ftrace?

- The official tracer for Linux (since 2008)
- Technically, “ftrace” is the way to attach callbacks to kernel functions.

vmware®
©2021 VMware, Inc.

What is ftrace?

- The official tracer for Linux (since 2008)
- Technically, “ftrace” is the way to attach callbacks to kernel functions.
- It is also known as the tracing system around /sys/kernel/tracing
  - (Formally in /sys/kernel/debug/tracing)

©2021 VMware, Inc.

How to interact with it?

```
# mount -t tracefs /sys/kernel/tracing
# cd /sys/kernel/tracing
# ls -F
```

```
available_events            max_graph_depth         stack_trace
available_filter_functions  options/                stack_trace_filter
available_tracers           per_cpu/                synthetic_events
buffer_percent              printk_formats          timestamp_mode
buffer_size_kb              README                  trace
buffer_total_size_kb        saved_cmdlines          trace_clock
current_tracer              saved_cmdlines_size     trace_marker
dynamic_events              saved_tgids             trace_marker_raw
dyn_ftrace_total_info       set_event               trace_options
enabled_functions           set_event_notrace_pid   trace_pipe
error_log                   set_event_pid           trace_stat/
eval_map                    set_ftrace_filter       tracing_cpumask
events/                     set_ftrace_notrace      tracing_max_latency
free_buffer                 set_ftrace_notrace_pid  tracing_on
function_profile_enabled    set_ftrace_pid          tracing_thresh
hwlat_detector/             set_graph_function      uprobe_events
instances/                  set_graph_notrace       uprobe_profile
kprobe_events               snapshot
kprobe_profile              stack_max_size
```

©2021 VMware, Inc.

How to interact with it?

```
# mount -t tracefs /sys/kernel/tracing
# cd /sys/kernel/tracing
# ls -F
```

```
available_events
available_filter_functions
available_tracers
buffer_percent
buffer_size_kb
buffer_total_size_kb
current_tracer
dynamic_events
dyn_ftrace_total_info
enabled_functions
error_log
eval_map
events/
free_buffer
function_profile_enabled
hwlat_detector/
instances/
kprobe_events
kprobe_profile
max_graph_depth
options/
per_cpu/
printk_formats
README
saved_cmdlines
saved_cmdlines_size
saved_tgids
set_event
set_event_notrace_pid
set_event_pid
set_ftrace_filter
set_ftrace_notrace
set_ftrace_notrace_pid
set_ftrace_pid
set_graph_function
set_graph_notrace
snapshot
stack_max_size
stack_trace
stack_trace_filter
synthetic_events
timestamp_mode
trace
trace_clock
trace_marker
trace_marker_raw
trace_options
trace_pipe
trace_stat/
tracing_cpumask
tracing_max_latency
tracing_on
tracing_thresh
uprobe_events
uprobe_profile
```

©2021 VMware, Inc.

What can be traced?

- Events
  - Static points in the kernel that provide various data

©2021 VMware, Inc.

What can be traced?

- Events
  - Static points in the kernel that provide various data
  - Can also be dynamic (kprobes)

vmware®
©2021 VMware, Inc.

What can be traced?

- Events
  - Static points in the kernel that provide various data
  - Can also be dynamic (kprobes)
- Tracers
  - Provides a specific functionality
    - Function tracing
    - Function Graph tracing
    - Latency tracing (interrupts disabled, etc)

vmware®
©2021 VMware, Inc.

Events

- Broken into groups
  - Scheduler
  - Interrupts

vmware®
©2021 VMware, Inc.

Events

- Broken into groups
  - Scheduler
  - Interrupts
- Traces data that the developer thinks is important
  - sched_switch (previous task → next task)
  - softirq raise (vector)

vmware®
©2021 VMware, Inc.

Enabling Events

```
# cd /sys/kernel/tracing
# echo 1 > events/sched/sched_switch/enable
# cat trace
```

```
# tracer: nop
#
# entries-in-buffer/entries-written: 285/285   #P:2
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
            bash-1263    [001] d... 108294.763885: sched_switch: prev_comm=bash prev_pid=1263 prev_prio=120 prev_state=R+ ==> next_comm=rcu_sched next_pid=12 next_prio=120
       rcu_sched-12      [001] d... 108294.764340: sched_switch: prev_comm=rcu_sched prev_pid=12 prev_prio=120 prev_state=I ==> next_comm=bash next_pid=1263 next_prio=120
          <idle>-0       [000] d... 108294.764414: sched_switch: prev_comm=swapper/0 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/u4:1 next_pid=13353 next_prio=120
            bash-1263    [001] d... 108294.764602: sched_switch: prev_comm=bash prev_pid=1263 prev_prio=120 prev_state=S ==> next_comm=sshd next_pid=1262 next_prio=120
    kworker/u4:1-13353   [000] d... 108294.764604: sched_switch: prev_comm=kworker/u4:1 prev_pid=13353 prev_prio=120 prev_state=I ==> next_comm=kworker/u4:0 next_pid=13473 next_
    kworker/u4:0-13473   [000] d... 108294.764608: sched_switch: prev_comm=kworker/u4:0 prev_pid=13473 prev_prio=120 prev_state=I ==> next_comm=swapper/0 next_pid=0 next_prio=120
            sshd-1262    [001] d... 108294.764675: sched_switch: prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108294.767451: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=rcu_sched next_pid=12 next_prio=120
       rcu_sched-12      [001] d... 108294.767454: sched_switch: prev_comm=rcu_sched prev_pid=12 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108294.770458: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:0 next_pid=7075 next_prio=120
     kworker/1:0-7075    [001] d... 108294.770547: sched_switch: prev_comm=kworker/1:0 prev_pid=7075 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108294.772409: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:0 next_pid=7075 next_prio=120
     kworker/1:0-7075    [001] d... 108294.772411: sched_switch: prev_comm=kworker/1:0 prev_pid=7075 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108294.957653: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:0 next_pid=7075 next_prio=120
     kworker/1:0-7075    [001] d... 108294.958026: sched_switch: prev_comm=kworker/1:0 prev_pid=7075 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108295.165590: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:0 next_pid=7075 next_prio=120
     kworker/1:0-7075    [001] d... 108295.165945: sched_switch: prev_comm=kworker/1:0 prev_pid=7075 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [000] d... 108295.197484: sched_switch: prev_comm=swapper/0 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kcompactd0 next_pid=27 next_prio=120
      kcompactd0-27      [000] d... 108295.198469: sched_switch: prev_comm=kcompactd0 prev_pid=27 prev_prio=120 prev_state=S ==> next_comm=kworker/0:0 next_pid=13277 next_prio=120
     kworker/0:0-13277   [000] d... 108295.199336: sched_switch: prev_comm=kworker/0:0 prev_pid=13277 prev_prio=120 prev_state=I ==> next_comm=swapper/0 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108295.269539: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:1H next_pid=197 next_prio=100
    kworker/1:1H-197     [001] d... 108295.269545: sched_switch: prev_comm=kworker/1:1H prev_pid=197 prev_prio=100 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108295.304303: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=sshd next_pid=1262 next_prio=120
            sshd-1262    [001] d... 108295.304383: sched_switch: prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
```

©2021 VMware, Inc.

Enabling Events

```
# cd /sys/kernel/tracing
# echo 1 > events/sched/sched_switch/enable
# cat trace
```

```
# tracer: nop
#
# entries-in-buffer/entries-written: 285/285   #P:2
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
            bash-1263    [001] d... 108294.763885: sched_switch: prev_comm=bash prev_pid=1263 prev_prio=120 prev_state=R+ ==> next_comm=rcu_sched next_pid=12 next_prio=120
       rcu_sched-12      [001] d... 108294.764340: sched_switch: prev_comm=rcu_sched prev_pid=12 prev_prio=120 prev_state=I ==> next_comm=bash next_pid=1263 next_prio=120
          <idle>-0       [000] d... 108294.764414: sched_switch: prev_comm=swapper/0 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/u4:1 next_pid=13353 next_prio=120
            bash-1263    [001] d... 108294.764602: sched_switch: prev_comm=bash prev_pid=1263 prev_prio=120 prev_state=S ==> next_comm=sshd next_pid=1262 next_prio=120
```

```
prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
```

```
     kworker/1:0-7075    [001] d... 108294.770547: sched_switch: prev_comm=kworker/1:0 prev_pid=7075 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108294.772409: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:0 next_pid=7075 next_prio=120
     kworker/1:0-7075    [001] d... 108294.772411: sched_switch: prev_comm=kworker/1:0 prev_pid=7075 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108294.957653: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:0 next_pid=7075 next_prio=120
     kworker/1:0-7075    [001] d... 108294.958026: sched_switch: prev_comm=kworker/1:0 prev_pid=7075 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108295.165590: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:0 next_pid=7075 next_prio=120
     kworker/1:0-7075    [001] d... 108295.165945: sched_switch: prev_comm=kworker/1:0 prev_pid=7075 prev_prio=120 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [000] d... 108295.197484: sched_switch: prev_comm=swapper/0 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kcompactd0 next_pid=27 next_prio=120
      kcompactd0-27      [000] d... 108295.198469: sched_switch: prev_comm=kcompactd0 prev_pid=27 prev_prio=120 prev_state=S ==> next_comm=kworker/0:0 next_pid=13277 next_prio=120
     kworker/0:0-13277   [000] d... 108295.199336: sched_switch: prev_comm=kworker/0:0 prev_pid=13277 prev_prio=120 prev_state=I ==> next_comm=swapper/0 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108295.269539: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=kworker/1:1H next_pid=197 next_prio=100
    kworker/1:1H-197     [001] d... 108295.269545: sched_switch: prev_comm=kworker/1:1H prev_pid=197 prev_prio=100 prev_state=I ==> next_comm=swapper/1 next_pid=0 next_prio=120
          <idle>-0       [001] d... 108295.304303: sched_switch: prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=sshd next_pid=1262 next_prio=120
            sshd-1262    [001] d... 108295.304383: sched_switch: prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
```

©2021 VMware, Inc.

Events Format

```
# cd /sys/kernel/tracing
# cat events/sched/sched_switch/format

name: sched_switch
ID: 315
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:char prev_comm[16];	offset:8;size:16;	signed:1;
	field:pid_t prev_pid;	offset:24;	size:4;	signed:1;
	field:int prev_prio;	offset:28;	size:4;	signed:1;
	field:long prev_state;	offset:32;	size:8;	signed:1;
	field:char next_comm[16];	offset:40;	size:16;	signed:1;
	field:pid_t next_pid;	offset:56;	size:4;	signed:1;
	field:int next_prio;	offset:60;	size:4;	signed:1;

print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

Events Format

```
prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
```

```
name: sched_switch
ID: 315
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:char prev_comm[16];   offset:8;size:16; signed:1;
     field:pid_t prev_pid;  offset:24;    size:4;  signed:1;
     field:int prev_prio;   offset:28;    size:4;  signed:1;
     field:long prev_state; offset:32;    size:8;  signed:1;
     field:char next_comm[16];   offset:40;    size:16; signed:1;
     field:pid_t next_pid;  offset:56;    size:4;  signed:1;
     field:int next_prio;   offset:60;    size:4;  signed:1;

print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

Events Format

```
prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
```

```
name: sched_switch
ID: 315
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:char prev_comm[16];   offset:8;size:16; signed:1;
        field:pid_t prev_pid;  offset:24;    size:4;  signed:1;
        field:int prev_prio;   offset:28;    size:4;  signed:1;
        field:long prev_state; offset:32;    size:8;  signed:1;
        field:char next_comm[16];   offset:40;    size:16; signed:1;
        field:pid_t next_pid;  offset:56;    size:4;  signed:1;
        field:int next_prio;   offset:60;    size:4;  signed:1;

print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

Events Format

```
prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
```

```
name: sched_switch
ID: 315
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:char prev_comm[16];   offset:8;size:16; signed:1;
        field:pid_t prev_pid;  offset:24;    size:4;  signed:1;
        field:int prev_prio;   offset:28;    size:4;  signed:1;
        field:long prev_state; offset:32;    size:8;  signed:1;
        field:char next_comm[16];   offset:40;    size:16; signed:1;
        field:pid_t next_pid;  offset:56;    size:4;  signed:1;
        field:int next_prio;   offset:60;    size:4;  signed:1;

print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

Events Format

```
prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
```

```
name: sched_switch
ID: 315
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:char prev_comm[16];   offset:8;size:16; signed:1;
        field:pid_t prev_pid;  offset:24;    size:4;  signed:1;
        field:int prev_prio;   offset:28;    size:4;  signed:1;
        field:long prev_state; offset:32;    size:8;  signed:1;
        field:char next_comm[16];   offset:40;    size:16; signed:1;
        field:pid_t next_pid;  offset:56;    size:4;  signed:1;
        field:int next_prio;   offset:60;    size:4;  signed:1;

print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

Events Format

```
prev_comm=sshd prev_pid=1262 prev_prio=120 prev_state=S ==> next_comm=swapper/1 next_pid=0 next_prio=120
```

```
name: sched_switch
ID: 315
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:char prev_comm[16];   offset:8;size:16; signed:1;
        field:pid_t prev_pid;  offset:24;    size:4;  signed:1;
        field:int prev_prio;   offset:28;    size:4;  signed:1;
        field:long prev_state; offset:32;    size:8;  signed:1;
        field:char next_comm[16];   offset:40;    size:16; signed:1;
        field:pid_t next_pid;  offset:56;    size:4;  signed:1;
        field:int next_prio;   offset:60;    size:4;  signed:1;

print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

Events Calling

kernel/sched/core.c:

```
static void __sched notrace __schedule(bool preempt)
{
    struct task_struct *prev, *next;
    unsigned long *switch_count;
    unsigned long prev_state;
    struct rq_flags rf;
    struct rq *rq;
    int cpu;
    [..]
        migrate_disable_switch(rq, prev);
        psi_sched_switch(prev, next, !task_on_rq_queued(prev));

        trace_sched_switch(preempt, prev, next);

        /* Also unlocks the rq: */
        rq = context_switch(rq, prev, next, &rf);
```

©2021 VMware, Inc.

Events Calling

kernel/sched/core.c:

```
static void __sched notrace __schedule(bool preempt)
{
    struct task_struct *prev, *next;
    unsigned long *switch_count;
    unsigned long prev_state;
    struct rq_flags rf;
    struct rq *rq;
    int cpu;
    [..]
        migrate_disable_switch(rq, prev);
        psi_sched_switch(prev, next, !task_on_rq_queued(prev));

        nop;

        /* Also unlocks the rq: */
        rq = context_switch(rq, prev, next, &rf);
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
TRACE_EVENT(sched_switch,

    TP_PROTO(bool preempt, struct task_struct *prev, struct task_struct *next),

    TP_ARGS(preempt, prev, next),

    TP_STRUCT__entry(
        __array(    char,   prev_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  prev_pid            )
        __field(    int,    prev_prio           )
        __field(    long,   prev_state          )
        __array(    char,   next_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  next_pid            )
        __field(    int,    next_prio           )
    ),

    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
                                                    trace_sched_switch(preempt, prev, next);
TRACE_EVENT(sched_switch,

    TP_PROTO(bool preempt, struct task_struct *prev, struct task_struct *next),

    TP_ARGS(preempt, prev, next),

    TP_STRUCT__entry(
        __array(    char,   prev_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  prev_pid            )
        __field(    int,    prev_prio           )
        __field(    long,   prev_state          )
        __array(    char,   next_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  next_pid            )
        __field(    int,    next_prio           )
    ),

    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
TRACE_EVENT(sched_switch,

    TP_PROTO(bool preempt, struct task_struct *prev, struct task_struct *next),

    TP_ARGS(preempt, prev, next),

    TP_STRUCT__entry(
        __array(    char,   prev_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  prev_pid            )
        __field(    int,    prev_prio           )
        __field(    long,   prev_state          )
        __array(    char,   next_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  next_pid            )
        __field(    int,    next_prio           )
    ),

    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
```

```
trace_sched_switch(preempt, prev, next);
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
TRACE_EVENT(sched_switch,

    TP_PROTO(bool preempt, struct task_struct *prev, struct task_struct *next),

    TP_ARGS(preempt, prev, next),

    TP_STRUCT__entry(
        __array(    char,   prev_comm,  TASK_COMM_LEN ) field:char prev_comm[16];   offset:8;   size:16;
        __field(    pid_t,  prev_pid            )     field:pid_t prev_pid;   offset:24;  size:4;
        __field(    int,    prev_prio           )     field:int prev_prio;    offset:28;  size:4;
        __field(    long,   prev_state          )     field:long prev_state;  offset:32;  size:8;
        __array(    char,   next_comm,  TASK_COMM_LEN ) field:char next_comm[16];   offset:40;  size:16;
        __field(    pid_t,  next_pid            )     field:pid_t next_pid;   offset:56;  size:4;
        __field(    int,    next_prio           )     field:int next_prio;    offset:60;  size:4;
    ),

    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
TRACE_EVENT(sched_switch,

    TP_PROTO(bool preempt, struct task_struct *prev, struct task_struct *next),

    TP_ARGS(preempt, prev, next),

    TP_STRUCT__entry(
        __array(    char,   prev_comm,  TASK_COMM_LEN ) field:char prev_comm[16];   offset:8;   size:16;
        __field(    pid_t,  prev_pid            )     field:pid_t prev_pid;   offset:24;  size:4;
        __field(    int,    prev_prio           )     field:int prev_prio;    offset:28;  size:4;
        __field(    long,   prev_state          )     field:long prev_state;  offset:32;  size:8;
        __array(    char,   next_comm,  TASK_COMM_LEN ) field:char next_comm[16];   offset:40;  size:16;
        __field(    pid_t,  next_pid            )     field:pid_t next_pid;   offset:56;  size:4;
        __field(    int,    next_prio           )     field:int next_prio;    offset:60;  size:4;
    ),

    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
TRACE_EVENT(sched_switch,

    TP_PROTO(bool preempt, struct task_struct *prev, struct task_struct *next),

    TP_ARGS(preempt, prev, next),

    TP_STRUCT__entry(
        __array(    char,   prev_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  prev_pid                    )
        __field(    int,    prev_prio                   )
        __field(    long,   prev_state                  )
        __array(    char,   next_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  next_pid                    )
        __field(    int,    next_prio                   )
    ),

    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN,,
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
```

```
struct {
    char    prev_comm[TASK_COMM_LEN];
    pid_t   prev_pid;
    int     prev_prio;
    long    prev_state;
    char    next_comm[TASK_COMM_LEN];
    pid_t   next_pid;
    int     next_prio;
} __entry;
```

vmware®
©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
        __array(    char,   next_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  next_pid            )
        __field(    int,    next_prio           )
    ),

    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
        __entry->prev_state = __trace_sched_switch_state(preempt, prev);
        memcpy(__entry->prev_comm, prev->comm, TASK_COMM_LEN);
        __entry->next_pid   = next->pid;
        __entry->next_prio  = next->prio;
        /* XXX SCHED_DEADLINE */
    ),

    TP_printk("prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s “
              ”next_pid=%d next_prio=%d",
        __entry->prev_comm, __entry->prev_pid, __entry->prev_prio,
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
        __array(    char,   next_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  next_pid            )
        __field(    int,    next_prio           )
    ),

    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
        __entry->prev_state = __trace_sched_switch_state(preempt, prev);
        memcpy(__entry->prev_comm, prev->comm, TASK_COMM_LEN);
        __entry->next_pid   = next->pid;
        __entry->next_prio  = next->prio;
        /* XXX SCHED_DEADLINE */
    ),

    TP_printk("prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s “
              ”next_pid=%d next_prio=%d",
        __entry->prev_comm, __entry->prev_pid, __entry->prev_prio,
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
        __array(    char,   next_comm,  TASK_COMM_LEN   )
        __field(    pid_t,  next_pid            )
        __field(    int,    next_prio           )
    ),
                                                trace_sched_switch(preempt, prev, next);
    TP_fast_assign(
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
        __entry->prev_state = __trace_sched_switch_state(preempt, prev);
        memcpy(__entry->prev_comm, prev->comm, TASK_COMM_LEN);
        __entry->next_pid   = next->pid;
        __entry->next_prio  = next->prio;
        /* XXX SCHED_DEADLINE */
    ),

    TP_printk("prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s “
              ”next_pid=%d next_prio=%d",
        __entry->prev_comm, __entry->prev_pid, __entry->prev_prio,
```

©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
void trace_sched_switch(bool preempt, stuct task_struct *prev, struct task_struct *next)
{
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
        __entry->prev_state = __trace_sched_switch_state(preempt, prev);
        memcpy(__entry->prev_comm, prev->comm, TASK_COMM_LEN);
        __entry->next_pid   = next->pid;
        __entry->next_prio  = next->prio;
        /* XXX SCHED_DEADLINE */
}
```

vmware®
©2021 VMware, Inc.

TRACE_EVENT Macro

include/trace/events/sched.h:

```
    TP_PROTO(bool preempt, struct task_struct *prev, struct task_struct *next),
```

```
void trace_sched_switch(bool preempt, stuct task_struct *prev, struct task_struct *next)
{
        memcpy(__entry->next_comm, next->comm, TASK_COMM_LEN);
        __entry->prev_pid   = prev->pid;
        __entry->prev_prio  = prev->prio;
        __entry->prev_state = __trace_sched_switch_state(preempt, prev);
        memcpy(__entry->prev_comm, prev->comm, TASK_COMM_LEN);
        __entry->next_pid   = next->pid;
        __entry->next_prio  = next->prio;
        /* XXX SCHED_DEADLINE */
}
```

vmware®
©2021 VMware, Inc.

TRACE_EVENT Macro

```
    TP_printk("prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s “
              ”next_pid=%d next_prio=%d",
        __entry->prev_comm, __entry->prev_pid, __entry->prev_prio,

        (__entry->prev_state & (TASK_REPORT_MAX - 1)) ?
          __print_flags(__entry->prev_state & (TASK_REPORT_MAX - 1), "|",
                { TASK_INTERRUPTIBLE, "S" },
                { TASK_UNINTERRUPTIBLE, "D" },
                { __TASK_STOPPED, "T" },
                { __TASK_TRACED, "t" },
                { EXIT_DEAD, "X" },
                { EXIT_ZOMBIE, "Z" },
                { TASK_PARKED, "P" },
                { TASK_DEAD, "I" }) :
          "R",

        __entry->prev_state & TASK_REPORT_MAX ? "+" : "",
        __entry->next_comm, __entry->next_pid, __entry->next_prio)
);
```

©2021 VMware, Inc.

TRACE_EVENT Macro

```
    TP_printk("prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s “
              ”next_pid=%d next_prio=%d",
        __entry->prev_comm, __entry->prev_pid, __entry->prev_prio,

        (__entry->prev_state & (TASK_REPORT_MAX - 1)) ?
          __print_flags(__entry->prev_state & (TASK_REPORT_MAX - 1), "|",
                { TASK_INTERRUPTIBLE, "S" },
                { TASK_UNINTERRUPTIBLE, "D" },
                { __TASK_STOPPED, "T" },
                { __TASK_TRACED, "t" },
                { EXIT_DEAD, "X" },
                { EXIT_ZOMBIE, "Z" },
                { TASK_PARKED, "P" },
                { TASK_DEAD, "I" }) :
          "R",

        __entry->prev_state & TASK_REPORT_MAX ? "+" : "",
        __entry->next_comm, __entry->next_pid, __entry->next_prio)
);
```

```
print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

TRACE_EVENT Macro

```
    TP_printk("prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s “
              ”next_pid=%d next_prio=%d",
        __entry->prev_comm, __entry->prev_pid, __entry->prev_prio,

        (__entry->prev_state & (TASK_REPORT_MAX - 1)) ?
          __print_flags(__entry->prev_state & (TASK_REPORT_MAX - 1), "|",
                { TASK_INTERRUPTIBLE, "S" },
                { TASK_UNINTERRUPTIBLE, "D" },
                { __TASK_STOPPED, "T" },
                { __TASK_TRACED, "t" },
                { EXIT_DEAD, "X" },
                { EXIT_ZOMBIE, "Z" },
                { TASK_PARKED, "P" },
                { TASK_DEAD, "I" }) :
          "R",

        __entry->prev_state & TASK_REPORT_MAX ? "+" : "",
        __entry->next_comm, __entry->next_pid, __entry->next_prio)
);
```

```
print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

TRACE_EVENT Macro

```
    TP_printk("prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s “
              ”next_pid=%d next_prio=%d",
        __entry->prev_comm, __entry->prev_pid, __entry->prev_prio,

        (__entry->prev_state & (TASK_REPORT_MAX - 1)) ?
          __print_flags(__entry->prev_state & (TASK_REPORT_MAX - 1), "|",
                { TASK_INTERRUPTIBLE, "S" },
                { TASK_UNINTERRUPTIBLE, "D" },
                { __TASK_STOPPED, "T" },
                { __TASK_TRACED, "t" },
                { EXIT_DEAD, "X" },
                { EXIT_ZOMBIE, "Z" },
                { TASK_PARKED, "P" },
                { TASK_DEAD, "I" }) :
          "R",

        __entry->prev_state & TASK_REPORT_MAX ? "+" : "",
        __entry->next_comm, __entry->next_pid, __entry->next_prio)
);
```

```
print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

TRACE_EVENT Macro

```
    TP_printk("prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s “
              ”next_pid=%d next_prio=%d",
        __entry->prev_comm, __entry->prev_pid, __entry->prev_prio,

        (__entry->prev_state & (TASK_REPORT_MAX - 1)) ?
          __print_flags(__entry->prev_state & (TASK_REPORT_MAX - 1), "|",
                { TASK_INTERRUPTIBLE, "S" },
                { TASK_UNINTERRUPTIBLE, "D" },
                { __TASK_STOPPED, "T" },
                { __TASK_TRACED, "t" },
                { EXIT_DEAD, "X" },
                { EXIT_ZOMBIE, "Z" },
                { TASK_PARKED, "P" },
                { TASK_DEAD, "I" }) :
          "R",

        __entry->prev_state & TASK_REPORT_MAX ? "+" : "",
        __entry->next_comm, __entry->next_pid, __entry->next_prio)
);
```

```
print fmt: "prev_comm=%s prev_pid=%d prev_prio=%d prev_state=%s%s ==> next_comm=%s next_pid=%d next_prio=%d",
REC->prev_comm, REC->prev_pid, REC->prev_prio, (REC->prev_state & ((((0x0000 | 0x0001 | 0x0002 | 0x0004 |
0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1)) ? __print_flags(REC->prev_state & ((((0x0000 | 0x0001 |
0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) - 1), "|", { 0x0001, "S" }, { 0x0002, "D" }, {
0x0004, "T" }, { 0x0008, "t" }, { 0x0010, "X" }, { 0x0020, "Z" }, { 0x0040, "P" }, { 0x0080, "I" }) : "R",
REC->prev_state & (((0x0000 | 0x0001 | 0x0002 | 0x0004 | 0x0008 | 0x0010 | 0x0020 | 0x0040) + 1) << 1) ? "+" :
"", REC->next_comm, REC->next_pid, REC->next_prio
```

©2021 VMware, Inc.

Event Examples

- You do not need to memorize all this
- Samples exist in `samples/trace_events`

vmware®
©2021 VMware, Inc.

Event Examples

- You do not need to memorize all this
- Samples exist in **`samples/trace_events`**
- The Makefile is important (has helpers as well)
  - see **`samples/trace_events/Makefile`**

vmware®
©2021 VMware, Inc.

Event Examples

- You do not need to memorize all this
- Samples exist in **`samples/trace_events`**
- The Makefile is important (has helpers as well)
  - see **`samples/trace_events/Makefile`**
- The TRACE_EVENT macro is in

  `samples/trace_events/trace-events-sample.h`
  - Well documented, with several examples of other helper functions
  - Examples of other special fields, like dynamic size strings

vmware®
©2021 VMware, Inc.

Event Examples

- You do not need to memorize all this
- Samples exist in **`samples/trace_events`**
- The Makefile is important (has helpers as well)
  - see **`samples/trace_events/Makefile`**
- The TRACE_EVENT macro is in

  `samples/trace_events/trace-events-sample.h`
  - Well documented, with several examples of other helper functions
  - Examples of other special fields, like dynamic size strings
- The calling of the macro is a kernel module in

  `samples/trace_events/trace-events-sample.c`

vmware®
©2021 VMware, Inc.

Tracers

- Defines functionality in the kernel
  - Trace all functions
  - Enable latency tracing

©2021 VMware, Inc.

Tracers

- Defines functionality in the kernel
  - Trace all functions
  - Enable latency tracing
- Events can be enabled within them

vmware®
©2021 VMware, Inc.

Tracers

- Defines functionality in the kernel
  - Trace all functions
  - Enable latency tracing
- Events can be enabled within them
- Some have their own options
  - Affects their functionality

©2021 VMware, Inc.

Tracers

```
# cd /sys/kernel/tracing
# echo function > current_tracer
# echo 1 > events/enable
# cat trace
```

```
# tracer: function
#
# entries-in-buffer/entries-written: 194418/2951438   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
          <idle>-0     [003] dN.. 72707.327393: rcu_read_lock_sched_held <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.327393: rcu_read_lock_held_common <-rcu_read_lock_sched_held
          <idle>-0     [003] dN.. 72707.327393: rcu_lockdep_current_cpu_online <-rcu_read_lock_held_common
          <idle>-0     [003] dN.. 72707.327394: rcu_qs <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.327394: rcu_utilization: End context switch
          <idle>-0     [003] dN.. 72707.327395: rcu_read_lock_sched_held <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.327395: rcu_read_lock_held_common <-rcu_read_lock_sched_held
          <idle>-0     [003] dN.. 72707.327395: rcu_lockdep_current_cpu_online <-rcu_read_lock_held_common
          <idle>-0     [003] dN.. 72707.327396: _raw_spin_lock <-__schedule
          <idle>-0     [003] dN.. 72707.327396: lock_acquire: 00000000423f8515 &rq->lock
          <idle>-0     [003] dN.. 72707.327397: do_raw_spin_lock <-__schedule
          <idle>-0     [003] dN.. 72707.327397: update_rq_clock <-__schedule
          <idle>-0     [003] dN.. 72707.327398: pick_next_task_fair <-__schedule
          <idle>-0     [003] dN.. 72707.327399: put_prev_task_idle <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327399: pick_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327399: clear_buddies <-pick_next_entity
          <idle>-0     [003] dN.. 72707.327400: set_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327401: __update_load_avg_se <-update_load_avg
          <idle>-0     [003] dN.. 72707.327401: __update_load_avg_cfs_rq <-update_load_avg
          <idle>-0     [003] dN.. 72707.327401: pick_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327402: clear_buddies <-pick_next_entity
          <idle>-0     [003] d... 72707.327406: sched_switch: prev_comm=swapper/3 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=sshd next_pid=3101 next_
          <idle>-0     [003] d... 72707.327407: rcu_read_lock_sched_held <-__schedule
```

©2021 VMware, Inc.

Tracers

```
# cd /sys/kernel/tracing
# echo function > current_tracer
# echo 1 > events/enable
# cat trace
```

```
# tracer: function
#
# entries-in-buffer/entries-written: 194418/2951438   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
          <idle>-0     [003] dN.. 72707.327393: rcu_read_lock_sched_held <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.327393: rcu_read_lock_held_common <-rcu_read_lock_sched_held
          <idle>-0     [003] dN.. 72707.327393: rcu_lockdep_current_cpu_online <-rcu_read_lock_held_common
          <idle>-0     [003] dN.. 72707.327394: rcu_qs <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.327394: rcu_utilization: End context switch
          <idle>-0     [003] dN.. 72707.327395: rcu_read_lock_sched_held <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.32739
          <idle>-0     [003] dN.. 72707.32739
          <idle>-0     [003] dN.. 72707.32739
          <idle>-0     [003] dN.. 72707.32739
          <idle>-0     [003] dN.. 72707.32739
          <idle>-0     [003] dN.. 72707.327397: update_rq_clock <-__schedule
          <idle>-0     [003] dN.. 72707.327398: pick_next_task_fair <-__schedule
          <idle>-0     [003] dN.. 72707.327399: put_prev_task_idle <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327399: pick_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327399: clear_buddies <-pick_next_entity
          <idle>-0     [003] dN.. 72707.327400: set_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327401: __update_load_avg_se <-update_load_avg
          <idle>-0     [003] dN.. 72707.327401: __update_load_avg_cfs_rq <-update_load_avg
          <idle>-0     [003] dN.. 72707.327401: pick_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327402: clear_buddies <-pick_next_entity
          <idle>-0     [003] d... 72707.327406: sched_switch: prev_comm=swapper/3 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=sshd next_pid=3101 next_
          <idle>-0     [003] d... 72707.327407: rcu_read_lock_sched_held <-__schedule
```

```
_raw_spin_lock <- __schedule
```

©2021 VMware, Inc.

Tracers (with events)

```
# cd /sys/kernel/tracing
# echo function > current_tracer
# echo 1 > events/enable
# cat trace
```

```
# tracer: function
#
# entries-in-buffer/entries-written: 194418/2951438   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
          <idle>-0     [003] dN.. 72707.327393: rcu_read_lock_sched_held <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.327393: rcu_read_lock_held_common <-rcu_read_lock_sched_held
          <idle>-0     [003] dN.. 72707.327393: rcu_lockdep_current_cpu_online <-rcu_read_lock_held_common
          <idle>-0     [003] dN.. 72707.327394: rcu_qs <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.327394: rcu_utilization: End context switch
          <idle>-0     [003] dN.. 72707.327395: rcu_read_lock_sched_held <-rcu_note_context_switch
          <idle>-0     [003] dN.. 72707.327395: rcu_read_lock_held_common <-rcu_read_lock_sched_held
          <idle>-0     [003] dN.. 72707.327395: rcu_lockdep_current_cpu_online <-rcu_read_lock_held_common
          <idle>-0     [003] dN.. 72707.327396: _raw_spin_lock <-__schedule
          <idle>-0     [003] dN.. 72707.327396: lock_acquire: 00000000423f8515 &rq->lock
          <idle>-0     [003] dN.. 72707.327397: do_raw_spin_lock <-__schedule
          <idle>-0     [003] dN.. 72707.327397: update_rq_clock <-__schedule
          <idle>-0     [003] dN.. 72707.327398: pick_next_task_fair <-__schedule
          <idle>-0     [003] dN.. 72707.327399: put_prev_task_idle <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327399: pick_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327399: clear_buddies <-pick_next_entity
          <idle>-0     [003] dN.. 72707.327400: set_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327401: __update_load_avg_se <-update_load_avg
          <idle>-0     [003] dN.. 72707.327401: __update_load_avg_cfs_rq <-update_load_avg
          <idle>-0     [003] dN.. 72707.327401: pick_next_entity <-pick_next_task_fair
          <idle>-0     [003] dN.. 72707.327402: clear_buddies <-pick_next_entity
          <idle>-0     [003] d... 72707.327406: sched_switch: prev_comm=swapper/3 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=sshd next_pid=3101 next_
          <idle>-0     [003] d... 72707.327407: rcu_read_lock_sched_held <-__schedule
```

©2021 VMware, Inc.

Tracer options

```
# cd /sys/kernel/tracing
# echo irqsoff > current_tracer
# cat trace
```

```
# tracer: irqsoff
#
# irqsoff latency trace v1.1.5 on 5.5.0-rc6-test+
# --------------------------------------------------------------------
# latency: 914 us, #85/85, CPU#0 | (M:desktop VP:0, KP:0, SP:0 HP:0 #P:4)
#    -----------------
#    | task: sshd-3101 (uid:0 nice:0 policy:0 rt_prio:0)
#    -----------------
#  => started at: interrupt_entry
#  => ended at:   restore_regs_and_return_to_kernel
#
#
#                  _------=> CPU#
#                 / _-----=> irqs-off
#                | / _----=> need-resched
#                || / _---=> hardirq/softirq
#                ||| / _--=> preempt-depth
#                |||| /     delay
#  cmd     pid   ||||| time  |   caller
#     \   /      |||||  \    |   /
    sshd-3101    0d...    1us : trace_hardirqs_off_thunk <-interrupt_entry
    sshd-3101    0d...    2us : do_IRQ <-ret_from_intr
    sshd-3101    0d...    2us : irq_enter <-do_IRQ
    sshd-3101    0d...    3us : rcu_irq_enter <-irq_enter
    sshd-3101    0d...    3us : irqtime_account_irq <-irq_enter
    sshd-3101    0d.h.    5us : handle_fasteoi_irq <-do_IRQ
    sshd-3101    0d.h.    5us : _raw_spin_lock <-handle_fasteoi_irq
    sshd-3101    0d.h.    8us : do_raw_spin_lock <-handle_fasteoi_irq
    sshd-3101    0d.h.    8us : irq_may_run <-handle_fasteoi_irq
    sshd-3101    0d.h.    9us : handle_irq_event <-handle_fasteoi_irq
    sshd-3101    0d.h.    9us : _raw_spin_unlock <-handle_irq_event
    sshd-3101    0d.h.   10us : do_raw_spin_unlock <-_raw_spin_unlock
    sshd-3101    0d.h.   10us : handle_irq_event_percpu <-handle_irq_event
    sshd-3101    0d.h.   11us : __handle_irq_event_percpu <-handle_irq_event_percpu
    sshd-3101    0d.h.   11us : rcu_read_lock_sched_held <-__handle_irq_event_percpu
```

©2021 VMware, Inc.

Tracer options

```
# cd /sys/kernel/tracing
# echo 0 > options/function-trace
# echo irqsoff > current_tracer
# cat trace
```

```
# tracer: irqsoff
#
# irqsoff latency trace v1.1.5 on 5.13.0-rc1-test+
# --------------------------------------------------------------------
# latency: 206952 us, #4/4, CPU#0 | (M:desktop VP:0, KP:0, SP:0 HP:0 #P:2)
#    -----------------
#    | task: swapper/0-0 (uid:0 nice:0 policy:0 rt_prio:0)
#    -----------------
#  => started at: irqentry_enter
#  => ended at:   __do_softirq
#
#
#                    _------=> CPU#
#                   / _-----=> irqs-off
#                  | / _----=> need-resched
#                  || / _---=> hardirq/softirq
#                  ||| / _--=> preempt-depth
#                  |||| /     delay
#  cmd     pid     ||||| time  |   caller
#     \   /        |||||  \    |   /
  <idle>-0         0d...    0us@: irqentry_enter
  <idle>-0         0d.s. 206951us : __do_softirq
  <idle>-0         0d.s. 206953us+: tracer_hardirqs_on <-__do_softirq
  <idle>-0         0d.s. 206982us : <stack trace>
 => __irq_exit_rcu
 => irq_exit_rcu
 => sysvec_apic_timer_interrupt
 => asm_sysvec_apic_timer_interrupt
 => native_safe_halt
 => default_idle
 => default_idle_call
 => do_idle
```

©2021 VMware, Inc.

Tracers (creating their own functionality)

```
# cd /sys/kernel/tracing
# echo hwlat > current_tracer
# cat trace
# tracer: hwlat
#
# entries-in-buffer/entries-written: 9/9   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
          <...>-3190  [001] d... 74129.667245: #1     inner/outer(us):  276/69    ts:1622384329.089336004
          <...>-3190  [002] d... 74130.673871: #2     inner/outer(us):   32/46    ts:1622384330.095961371
          <...>-3190  [003] d... 74131.681929: #3     inner/outer(us):   48/38    ts:1622384331.104018173
          <...>-3190  [000] d... 74132.690079: #4     inner/outer(us):   34/52    ts:1622384332.112167460
          <...>-3190  [001] d... 74133.698163: #5     inner/outer(us):  232/38    ts:1622384333.120250818
          <...>-3190  [000] d... 74136.722049: #8     inner/outer(us):  127/114   ts:1622384336.144133791
          <...>-3190  [001] d... 74137.730126: #9     inner/outer(us):   24/295   ts:1622384337.152210203
          <...>-3190  [002] d... 74138.737613: #10    inner/outer(us):   54/27    ts:1622384338.159696113
          <...>-3190  [003] d... 74139.745649: #11    inner/outer(us):   31/124   ts:1622384339.167731440
```

©2021 VMware, Inc.

Instances

- More than one trace buffer

©2021 VMware, Inc.

Instances

- More than one trace buffer
- Allows for more than one tracer at a time

©2021 VMware, Inc.

Instances

- More than one trace buffer
- Allows for more than one tracer at a time
- Prevent one event from being overwritten by other more active events

©2021 VMware, Inc.

Instances

- More than one trace buffer
- Allows for more than one tracer at a time
- Prevent one event from being overwritten by other more active events

  /sys/kernel/tracing/instances/

- Simply create with **mkdir**

```
# mkdir instances/foo
# ls instances/foo

available_tracers     free_buffer          set_ftrace_pid    trace_options
buffer_percent        options              snapshot          trace_pipe
buffer_size_kb        per_cpu              timestamp_mode    tracing_cpumask
buffer_total_size_kb  set_event            trace             tracing_max_latency
current_tracer        set_event_pid        trace_clock       tracing_on
error_log             set_ftrace_filter    trace_marker
events                set_ftrace_notrace   trace_marker_raw
```

vmware®
©2021 VMware, Inc.

Instances

```
# cd /sys/kernel/tracing
# echo function > current_tracer
# mkdir instances/foo
# cd instances/foo
# echo 1 > events/enable
# cat trace
```

```
# tracer: nop
#
# entries-in-buffer/entries-written: 293282/3333177   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
          <idle>-0       [003] d..1   717.463304: cpu_idle: state=4294967295 cpu_id=3
          <idle>-0       [003] d..1   717.463306: irq_enable: caller=cpuidle_enter_state+0xdd/0x300 parent=0x0
          <idle>-0       [003] d..1   717.463308: irq_disable: caller=irqentry_enter+0x4a/0x60 parent=0x0
          <idle>-0       [003] d.h1   717.463310: lock_acquire: 000000008d44ca53 read tk_core.seq.seqcount
          <idle>-0       [003] d.h1   717.463310: lock_release: 000000008d44ca53 tk_core.seq.seqcount
          <idle>-0       [003] d.h1   717.463311: call_function_entry: vector=252
          <idle>-0       [003] d.h1   717.463313: call_function_exit: vector=252
          <idle>-0       [003] d..1   717.463316: lock_acquire: 000000008d44ca53 read tk_core.seq.seqcount
          <idle>-0       [003] d..1   717.463316: lock_release: 000000008d44ca53 tk_core.seq.seqcount
          <idle>-0       [003] d..1   717.463316: irq_enable: caller=irqentry_exit+0x5c/0x80 parent=0x0
          <idle>-0       [003] d..1   717.463320: irq_disable: caller=do_idle+0xbb/0x2e0 parent=0x0
```

©2021 VMware, Inc.

Tracers vs Events

- Events:
  - Are static points in the kernel
  - Gives specific information on data at the time of execution

vmware®
©2021 VMware, Inc.

Tracers vs Events

- Events:
  - Are static points in the kernel
  - Gives specific information on data at the time of execution
- Tracers:
  - Add functionality to how the things are traced
  - Have their own options
  - Can include events in the recordings

vmware®
©2021 VMware, Inc.

Dynamic Events

- Same as normal events

©2021 VMware, Inc.

Dynamic Events

- Same as normal events
- Are not created by the TRACE_EVENT macro

vmware®
©2021 VMware, Inc.

Dynamic Events

- Same as normal events
- Are not created by the TRACE_EVENT macro
- Injected via a break point or from another event
  - kprobes - break point (or even function tracer)
  - synthetic events - events related to more than one event

vmware®
©2021 VMware, Inc.

Kprobe Events

- Described in

  `Documentation/trace/kprobetrace.rst`

vmware®
©2021 VMware, Inc.

Kprobe Events

- Described in

  `Documentation/trace/kprobetrace.rst`

- Interface in

  `/sys/kernel/tracing/kprobe_events`

©2021 VMware, Inc.

Kprobe Events

- Described in

  `Documentation/trace/kprobetrace.rst`

- Interface in

  `/sys/kernel/tracing/kprobe_events`

- Can attach directly to functions
  - Uses ftrace function callback interface (fast)

vmware®
©2021 VMware, Inc.

Kprobe Events

- Described in

  `Documentation/trace/kprobetrace.rst`

- Interface in

  `/sys/kernel/tracing/kprobe_events`

- Can attach directly to functions
  - Uses ftrace function callback interface (fast)
  - Or indexed (breakpoints - slow)
    - But can be “optimized” with jmps in some locations (fast)

vmware®
©2021 VMware, Inc.

Kprobe Events (trace activate_task task ‘p’)

kernel/sched/core.c:

```
void activate_task(struct rq *rq, struct task_struct *p, int flags)
{
    enqueue_task(rq, p, flags);

    p->on_rq = TASK_ON_RQ_QUEUED;
}
```

©2021 VMware, Inc.

Kprobe Events (get task ‘p’s pid and comm)

include/linux/sched.h:

```
struct task_struct {
        [..]
        pid_t                   pid;
        [..]
        char                    comm[TASK_COMM_LEN];
        [..]
```

©2021 VMware, Inc.

Kprobe Events (gdb trick)

```
$ gdb vmlinux
(gdb) p &((struct task_struct *)0)->pid)
$1 = (pid_t *) 0x888
(gdb) p &((struct task_struct *)0)->comm)
$2 = (char (*)[16]) 0xac8
```

©2021 VMware, Inc.

Kprobe Events (gdb trick)

```
$ gdb vmlinux
(gdb) p &((struct task_struct *)0)->pid)
$1 = (pid_t *) 0x888
(gdb) p &((struct task_struct *)0)->comm)
$2 = (char (*)[16]) 0xac8
(gdb) quit
$ su
# cd /sys/kernel/tracing
# echo ‘p:activate activate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' > kprobe_events
```

©2021 VMware, Inc.

Kprobe Events

```
#
# echo 1 > events/kprobes/activate
# cat trace

# tracer: nop
#
# entries-in-buffer/entries-written: 9633/9633   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
             bash-3102  [002] d... 77050.054408: activate: (activate_task+0x0/0xf0) pid=3274 comm="kworker/u8:1"
          <idle>-0     [000] d.h. 77050.054592: activate: (activate_task+0x0/0xf0) pid=3101 comm="sshd"
             bash-3102  [002] d... 77050.054804: activate: (activate_task+0x0/0xf0) pid=3274 comm="kworker/u8:1"
          <idle>-0     [001] d.h. 77050.055085: activate: (activate_task+0x0/0xf0) pid=10 comm="rcu_sched"
             sshd-3101  [000] d.s. 77050.056998: activate: (activate_task+0x0/0xf0) pid=3176 comm="kworker/0:0"
          <idle>-0     [001] d.s. 77050.059078: activate: (activate_task+0x0/0xf0) pid=10 comm="rcu_sched"
          <idle>-0     [001] d.s. 77050.063072: activate: (activate_task+0x0/0xf0) pid=10 comm="rcu_sched"
          <idle>-0     [001] d.s. 77050.067093: activate: (activate_task+0x0/0xf0) pid=10 comm="rcu_sched"
          <idle>-0     [001] d.s. 77050.071060: activate: (activate_task+0x0/0xf0) pid=10 comm="rcu_sched"
          <idle>-0     [003] d.s. 77050.088144: activate: (activate_task+0x0/0xf0) pid=2692 comm="kworker/3:1"
          <idle>-0     [000] d.s. 77050.264143: activate: (activate_task+0x0/0xf0) pid=3176 comm="kworker/0:0"
          <idle>-0     [001] d.h. 77050.265197: activate: (activate_task+0x0/0xf0) pid=10 comm="rcu_sched"
          <idle>-0     [001] d.s. 77050.269123: activate: (activate_task+0x0/0xf0) pid=10 comm="rcu_sched"
          <idle>-0     [001] d.s. 77050.273089: activate: (activate_task+0x0/0xf0) pid=10 comm="rcu_sched"
```

©2021 VMware, Inc.

Synthetic Events and Histograms

- Described in

  `Documentation/trace/histogram.rst`

©2021 VMware, Inc.

Synthetic Events and Histograms

- Described in

  Documentation/trace/histogram.rst

- Interface in

  /sys/kernel/tracing/synthetic_events

  /sys/kernel/tracing/events/*/*/trigger

  /sys/kernel/tracing/events/*/*/hist

vmware®
©2021 VMware, Inc.

Synthetic Events and Histograms

- Described in

  `Documentation/trace/histogram.rst`

- Interface in

  `/sys/kernel/tracing/synthetic_events`

  `/sys/kernel/tracing/events/*/*/trigger`

  `/sys/kernel/tracing/events/*/*/hist`

- Creates an event based on two other events

vmware®
©2021 VMware, Inc.

Synthetic Events and Histograms

- Described in

  `Documentation/trace/histogram.rst`

- Interface in

  `/sys/kernel/tracing/synthetic_events`

  `/sys/kernel/tracing/events/*/*/trigger`

  `/sys/kernel/tracing/events/*/*/hist`

- Creates an event based on two other events
- Can track the time between the events (latency)

vmware®
©2021 VMware, Inc.

Synthetic Events and Histograms

```
# cd /sys/kernel/tracing
# echo ‘p:activate activate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' > kprobe_events
# echo ‘p:deactivate deactivate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' >> kprobe_events

# echo ‘hist:keys=pid:ts=common_timestamp.usec' > events/kprobes/deactivate/trigger
```

©2021 VMware, Inc.

Synthetic Events and Histograms

```
# cd /sys/kernel/tracing
# echo 'p:activate activate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' > kprobe_events
# echo 'p:deactivate deactivate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' >> kprobe_events

# echo 'hist:keys=pid:ts=common_timestamp.usec' > events/kprobes/deactivate/trigger
-bash: echo: write error: Invalid argument
```

©2021 VMware, Inc.

Synthetic Events and Histograms

```
# cd /sys/kernel/tracing
# echo 'p:activate activate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' > kprobe_events
# echo 'p:deactivate deactivate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' >> kprobe_events

# echo 'hist:keys=pid:ts=common_timestamp.usec' > events/kprobes/deactivate/trigger
-bash: echo: write error: Invalid argument

# cat error_log
[80364.243786] hist:kprobes:deactivate: error: Invalid field modifier
  Command: keys=pid:ts=common_timestamp.usec
                                         ^
```

©2021 VMware, Inc.

Synthetic Events and Histograms

```
# cd /sys/kernel/tracing
# echo 'p:activate activate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' > kprobe_events
# echo 'p:deactivate deactivate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' >> kprobe_events

# echo 'hist:keys=pid:ts=common_timestamp.usec' > events/kprobes/deactivate/trigger
-bash: echo: write error: Invalid argument

# cat error_log
[80364.243786] hist:kprobes:deactivate: error: Invalid field modifier
  Command: keys=pid:ts=common_timestamp.usec
                                         ^

# echo 'hist:keys=pid:ts=common_timestamp.usecs' > events/kprobes/deactivate/trigger
```

©2021 VMware, Inc.

Synthetic Events and Histograms

```
# cd /sys/kernel/tracing
# echo 'p:activate activate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' > kprobe_events
# echo 'p:deactivate deactivate_task pid=+0x888($arg2):u32 comm=+0xac8($arg2):string' >> kprobe_events

# echo 'hist:keys=pid:ts=common_timestamp.usec' > events/kprobes/deactivate/trigger
-bash: echo: write error: Invalid argument

# cat error_log
[80364.243786] hist:kprobes:deactivate: error: Invalid field modifier
  Command: keys=pid:ts=common_timestamp.usec
                                         ^

# echo 'hist:keys=pid:ts=common_timestamp.usecs' > events/kprobes/deactivate/trigger

# echo 'activate_lat u32 pid; u64 lat;' > synthetic_events

# echo 'hist:keys=pid:lat=common_timestamp.usecs-$ts:onmatch(kprobes.deactivate)'\
  '.trace(activate_lat,pid,$lat)' > events/kprobes/activate/trigger
```

©2021 VMware, Inc.

Synthetic Events and Histograms

```
# echo 1 > events/synthetic/activate_lat/enable
# cat trace

# tracer: nop
#
# entries-in-buffer/entries-written: 286/286   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
          <idle>-0     [000] d.h. 81050.285616: activate_lat: pid=3491 lat=100019
    kworker/u8:2-3491  [000] d... 81050.285669: activate_lat: pid=3101 lat=81050285665
            sshd-3101  [000] d.h. 81050.285791: activate_lat: pid=3491 lat=100
    kworker/u8:2-3491  [000] d... 81050.285836: activate_lat: pid=3101 lat=20
          <idle>-0     [002] d.h. 81050.286203: activate_lat: pid=10 lat=18909
          <idle>-0     [002] d.s. 81050.290229: activate_lat: pid=10 lat=3981
          <idle>-0     [002] d.s. 81050.294227: activate_lat: pid=10 lat=3945
          <idle>-0     [000] d.s. 81050.410364: activate_lat: pid=3176 lat=81050410333
          <idle>-0     [003] d.h. 81050.410781: activate_lat: pid=2692 lat=432501
          <idle>-0     [001] d.h. 81050.412110: activate_lat: pid=3259 lat=3761880
          <idle>-0     [002] d.h. 81050.412401: activate_lat: pid=10 lat=118122
          <idle>-0     [002] d.s. 81050.416227: activate_lat: pid=10 lat=3743
          <idle>-0     [002] dNs. 81050.416261: activate_lat: pid=3342 lat=81050416255
          <idle>-0     [002] d.s. 81050.420357: activate_lat: pid=10 lat=3988
          <idle>-0     [001] d.s. 81050.423428: activate_lat: pid=3259 lat=10249
          <idle>-0     [002] d.h. 81050.424284: activate_lat: pid=10 lat=3839
```

©2021 VMware, Inc.

Synthetic Events and Histograms

```
# echo 'hist:keys=pid,lat if lat < 10000' > events/synthetic/activate_lat/trigger
# cat events/synthetic/activate_lat/hist

# event histogram
#
# trigger info: hist:keys=pid,lat:vals=hitcount:sort=hitcount:size=2048 if lat < 10000 [active]
#

{ pid:       3101, lat:        195 } hitcount:          1
{ pid:         10, lat:       5023 } hitcount:          1
{ pid:       3101, lat:        503 } hitcount:          1
{ pid:       3101, lat:        702 } hitcount:          1
{ pid:       3780, lat:        381 } hitcount:          1
{ pid:         10, lat:       3844 } hitcount:          1
{ pid:       3780, lat:        460 } hitcount:          1
{ pid:         10, lat:       3908 } hitcount:          1
{ pid:       3101, lat:        153 } hitcount:          1
{ pid:       3780, lat:         82 } hitcount:          1
{ pid:         10, lat:       3993 } hitcount:          1
{ pid:         10, lat:       3821 } hitcount:          1
{ pid:         10, lat:       3815 } hitcount:          1
{ pid:         10, lat:       3849 } hitcount:          1
{ pid:         10, lat:       3932 } hitcount:          1
{ pid:         10, lat:       3995 } hitcount:          1
{ pid:         10, lat:       3916 } hitcount:          1
{ pid:         10, lat:       3890 } hitcount:          1
{ pid:         10, lat:       4048 } hitcount:          1
{ pid:         10, lat:       4062 } hitcount:          1
{ pid:       3780, lat:        584 } hitcount:          1

Totals:
    Hits: 21
    Entries: 21
    Dropped: 0
```

©2021 VMware, Inc.

Debugging the kernel

- trace_printk()
  - Just like printk() but writes to the ring buffer

vmware®
©2021 VMware, Inc.

Debugging the kernel

- trace_printk()
  - Just like printk() but writes to the ring buffer
  - Can be used in any context (Interrupts, scheduler, even NMIs)

vmware®
©2021 VMware, Inc.

Debugging the kernel

- trace_printk()
  - Just like printk() but writes to the ring buffer
  - Can be used in any context (Interrupts, scheduler, even NMIs)
  - Should not be left in production
    - Use TRACE_EVENT() for that

vmware®
©2021 VMware, Inc.

Debugging the kernel

- trace_printk()
  - Just like printk() but writes to the ring buffer
  - Can be used in any context (Interrupts, scheduler, even NMIs)
  - **Should not be left in production**

```
**********************************************************
**   NOTICE NOTICE NOTICE NOTICE NOTICE NOTICE NOTICE   **
**                                                      **
** trace_printk() being used. Allocating extra memory.  **
**                                                      **
** This means that this is a DEBUG kernel and it is     **
** unsafe for production use.                           **
**                                                      **
** If you see this message and you are not debugging    **
** the kernel, report this immediately to your vendor!  **
**                                                      **
**   NOTICE NOTICE NOTICE NOTICE NOTICE NOTICE NOTICE   **
**********************************************************
```

©2021 VMware, Inc.

Debugging the kernel

```
diff --git a/kernel/sched/core.c b/kernel/sched/core.c
index 5226cc26a095..7efea5141f82 100644
--- a/kernel/sched/core.c
+++ b/kernel/sched/core.c
@@ -1620,6 +1620,7 @@ static inline void dequeue_task(struct rq *rq, struct task_struct *p, int flags)

 void activate_task(struct rq *rq, struct task_struct *p, int flags)
 {
+       trace_printk("pid=%d comm=%s\n", p->pid, p->comm);
        enqueue_task(rq, p, flags);

        p->on_rq = TASK_ON_RQ_QUEUED;
```

©2021 VMware, Inc.

Debugging the kernel

```
# cd /sys/kernel/tracing
# cat trace

# tracer: nop
#
# entries-in-buffer/entries-written: 158246/165969   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
          systemd-1       [004] d..3    10.896871: activate_task: pid=2 comm=kthreadd
          kauditd-68      [005] d..5    10.896942: activate_task: pid=1 comm=swapper/0
          systemd-1       [004] d..3    10.896953: activate_task: pid=68 comm=kauditd
          systemd-1       [004] d.s5    10.905596: activate_task: pid=14 comm=rcu_preempt
##### CPU 0 buffer started ####
           <idle>-0       [000] d.s4    10.908777: activate_task: pid=14 comm=rcu_preempt
           <idle>-0       [000] d.s4    10.912807: activate_task: pid=14 comm=rcu_preempt
           <idle>-0       [000] d.s4    10.916833: activate_task: pid=14 comm=rcu_preempt
           <idle>-0       [000] d.s4    10.920862: activate_task: pid=14 comm=rcu_preempt
          systemd-1       [004] d..4    10.923951: activate_task: pid=58 comm=kworker/4:1
           <idle>-0       [000] d.s4    10.924893: activate_task: pid=14 comm=rcu_preempt
           <idle>-0       [000] d.s4    10.928922: activate_task: pid=14 comm=rcu_preempt
          systemd-1       [004] d.s5    10.931308: activate_task: pid=14 comm=rcu_preempt
          systemd-1       [004] d..4    10.935521: activate_task: pid=58 comm=kworker/4:1
          systemd-1       [004] d.s5    10.939543: activate_task: pid=58 comm=kworker/4:1
          systemd-1       [004] d.s5    10.947345: activate_task: pid=58 comm=kworker/4:1
```

©2021 VMware, Inc.

Tools and Libraries

- trace-cmd - Interface to tracefs and records traces to a file (trace.dat)

  https://www.trace-cmd.org/

vmware®
©2021 VMware, Inc.

Tools and Libraries

- trace-cmd - Interface to tracefs and records traces to a file (trace.dat)

  https://www.trace-cmd.org/

- KernelShark - GUI interface that can read trace.dat files (and more)

  https://www.kernelshark.org/

vmware®
©2021 VMware, Inc.

Tools and Libraries

- trace-cmd - Interface to tracefs and records traces to a file (trace.dat)

  https://www.trace-cmd.org/

- KernelShark - GUI interface that can read trace.dat files (and more)

  https://www.kernelshark.org/

- libtracecmd - library to read and write to the trace.dat file

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

vmware®
©2021 VMware, Inc.

Tools and Libraries

- trace-cmd - Interface to tracefs and records traces to a file (trace.dat)

  https://www.trace-cmd.org/

- KernelShark - GUI interface that can read trace.dat files (and more)

  https://www.kernelshark.org/

- libtracecmd - library to read and write to the trace.dat file

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

- libtracefs - library to access the tracefs directory (this presentation)

  https://git.kernel.org/pub/scm/libs/libtrace/libtracefs.git/

vmware®
©2021 VMware, Inc.

Tools and Libraries

- trace-cmd - Interface to tracefs and records traces to a file (trace.dat)

  https://www.trace-cmd.org/

- KernelShark - GUI interface that can read trace.dat files (and more)

  https://www.kernelshark.org/

- libtracecmd - library to read and write to the trace.dat file

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

- libtracefs - library to access the tracefs directory (this presentation)

  https://git.kernel.org/pub/scm/libs/libtrace/libtracefs.git/

- libtraceevent - library to parse the raw event fields
  - Uses the event format files (offsets and sizes)

    https://git.kernel.org/pub/scm/libs/libtrace/libtraceevent.git/

vmware®
©2021 VMware, Inc.

Debugging Userspace interaction with the kernel

- `/sys/kernel/tracing/`**`trace_marker`**

©2021 VMware, Inc.

Debugging Userspace interaction with the kernel

- `/sys/kernel/tracing/`**`trace_marker`**
- Used to write into the ring buffer from user space

vmware®
©2021 VMware, Inc.

Debugging Userspace interaction with the kernel

- `/sys/kernel/tracing/`**`trace_marker`**
- Used to write into the ring buffer from user space
- Open it at the start of your application
  - Write to it in your application
  - See where the application was within the events

vmware®
©2021 VMware, Inc.

Debugging Userspace interaction with the kernel

- `/sys/kernel/tracing/`**`trace_marker`**
- Used to write into the ring buffer from user space
- Open it at the start of your application
  - Write to it in your application
  - See where the application was within the events
- libtracefs has helper functions

```
tracefs_print_init()
tracefs_printf()
tracefs_vprintf()
tracefs_print_close()
```

vmware®
©2021 VMware, Inc.

trace_marker (access from command line)

```
# cd /sys/kernel/tracing
# echo hello top level > trace_marker
# mkdir instances/foo
# echo hello foo > instance/foo/trace_marker
# cat trace

# tracer: nop
#
# entries-in-buffer/entries-written: 1/1   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
          <...>-2383    [002] ....  3989.180768: tracing_mark_write: hello top level

# cat instances/foo/trace

# tracer: nop
#
# entries-in-buffer/entries-written: 1/1   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
          <...>-2383    [002] ....  3970.164828: tracing_mark_write: hello foo
```

©2021 VMware, Inc.

libtracefs (simple-trace-print.c)

```c
#include <unistd.h>
#include <tracefs.h>

int main (int argc, char **argv, char **env)
{
    struct tracefs_instance *instance;

    instance = tracefs_instance_create("my-buffer");
    if (!instance)
        return -1;

    tracefs_print_init(NULL);
    tracefs_print_init(instance);
    tracefs_printf(NULL, "Enabing events");
    tracefs_event_enable(instance, "sched", NULL);
    tracefs_printf(instance, "Go to sleep");
    sleep(1);
    tracefs_printf(instance, "Wake up!");
    tracefs_event_disable(instance, "sched", NULL);
    tracefs_printf(NULL, "Disable events");

    /* Want to see the instance after this */
    // tracefs_instance_destroy(instance);

    tracefs_print_close(instance);
    tracefs_print_close(NULL);
    return 0;
}
```

vmware®
©2021 VMware, Inc.

libtracefs

```
# ./simple-trace-print
# trace-cmd show

# tracer: nop
#
# entries-in-buffer/entries-written: 2/2   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
 simple-trace-pr-2147    [000] ....  1480.936742: tracing_mark_write: Enabing events
 simple-trace-pr-2147    [000] ....  1481.940732: tracing_mark_write: Disable events
```

©2021 VMware, Inc.

libtracefs

```
# trace-cmd show -B my-buffer

# tracer: nop
#
# entries-in-buffer/entries-written: 155/155   #P:8
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| /     delay
#           TASK-PID     CPU#  ||||   TIMESTAMP  FUNCTION
#              | |         |   ||||      |         |
 simple-trace-pr-2147    [000] d.h2  1480.938406: sched_stat_runtime: comm=simple-trace-pr pid=2147 runtime=1006385 [ns] vruntime=1031928670 [ns]
 simple-trace-pr-2147    [000] ....  1480.938747: tracing_mark_write: Go to sleep
 simple-trace-pr-2147    [000] d..2  1480.938751: sched_stat_runtime: comm=simple-trace-pr pid=2147 runtime=346454 [ns] vruntime=1032275124 [ns]
 simple-trace-pr-2147    [000] d..2  1480.938754: sched_switch: prev_comm=simple-trace-pr prev_pid=2147 prev_prio=120 prev_state=S ==> next_comm=swapper/0 next_pid=0 next
          <idle>-0       [007] d.s3  1480.941409: sched_waking: comm=rcu_preempt pid=14 prio=120 target_cpu=007
          <idle>-0       [007] dNs4  1480.941415: sched_wakeup: comm=rcu_preempt pid=14 prio=120 target_cpu=007
[..]
          <idle>-0       [000] dNh4  1481.938809: sched_wakeup: comm=simple-trace-pr pid=2147 prio=120 target_cpu=000
          <idle>-0       [000] d..2  1481.938817: sched_switch: prev_comm=swapper/0 prev_pid=0 prev_prio=120 prev_state=R ==> next_comm=simple-trace-pr next_pid=2147 next
 simple-trace-pr-2147    [000] ....  1481.938830: tracing_mark_write: Wake up!
 simple-trace-pr-2147    [000] d.h1  1481.939388: sched_stat_runtime: comm=simple-trace-pr pid=2147 runtime=582677 [ns] vruntime=1032857801 [ns]
 simple-trace-pr-2147    [000] d.h1  1481.939391: sched_wake_idle_without_ipi: cpu=1
 simple-trace-pr-2147    [000] d.s4  1481.940391: sched_waking: comm=rcu_preempt pid=14 prio=120 target_cpu=007
 simple-trace-pr-2147    [000] d.s5  1481.940393: sched_wakeup: comm=rcu_preempt pid=14 prio=120 target_cpu=007
```

©2021 VMware, Inc.

Thank You

Confidential | ©2021 VMware, Inc.