What’s new in Ftrace

And what’s new to you!

What is “ftrace”

- The official tracer of the Linux kernel

What is “ftrace”

- The official tracer of the Linux kernel
- Added to Linux in 2.6.27 in 2008

What is “ftrace”

- The official tracer of the Linux kernel
- Added to Linux in 2.6.27 in 2008
- “Ftrace” really is the “function tracer”
  - But also used for the infrastructure that houses the function tracer

What is “ftrace”

- The official tracer of the Linux kernel
- Added to Linux in 2.6.27 in 2008
- “Ftrace” really is the “function tracer”
  - But also used for the infrastructure that houses the function tracer
- Was designed to be easily used in embedded environments
  - Works with just busybox (cat and echo commands)

What is “ftrace”

- The official tracer of the Linux kernel
- Added to Linux in 2.6.27 in 2008
- “Ftrace” really is the “function tracer”
  - But also used for the infrastructure that houses the function tracer
- Was designed to be easily used in embedded environments
  - Works with just busybox (cat and echo commands)
- If you need to know more
  - Watch the videos from here: https://kernel-recipes.org

Why this talk?

Why this talk?

- I mentioned a feature on twitter

Why this talk?

- I mentioned a feature on twitter
  - Someone said “wow, didn’t know. Great feature”

Why this talk?

- I mentioned a feature on twitter
  - Someone said “wow, didn’t know. Great feature”
    - “Never heard of it”

Why this talk?

- I mentioned a feature on twitter
  - Someone said “wow, didn’t know. Great feature”
    - “Never heard of it”
    - “I’ve been using ftrace for years!”

Why this talk?

- I mentioned a feature on twitter
  - Someone said “wow, didn’t know. Great feature”
    - “Never heard of it”
    - “I’ve been using ftrace for years!”
- I decided to add a new feature

Why this talk?

- I mentioned a feature on twitter
  - Someone said “wow, didn’t know. Great feature”
    - “Never heard of it”
    - “I’ve been using ftrace for years!”
- I decided to add a new feature
  - Realized it already existed

Why this talk?

- I mentioned a feature on twitter
  - Someone said “wow, didn’t know. Great feature”
    - “Never heard of it”
    - “I’ve been using ftrace for years!”
- I decided to add a new feature
  - Realized it already existed
  - I wrote it!

Why this talk?

- I mentioned a feature on twitter
  - Someone said “wow, didn’t know. Great feature”
    - “Never heard of it”
    - “I’ve been using ftrace for years!”
- I decided to add a new feature
  - Realized it already existed
  - I wrote it!
- I need to write a book

What’s new?

What’s new? (to you!)

What’s new? (to you!)

- Kprobe trace (2009)

What’s new? (to you!)

- Kprobe trace (2009)
  - A lot of people still do not know about this

What’s new? (to you!)

- Kprobe trace (2009)
  - A lot of people still do not know about this
  - Dynamically create trace events almost anywhere in the kernel

What’s new? (to you!)

- Kprobe trace (2009)
  - A lot of people still do not know about this
  - Dynamically create trace events almost anywhere in the kernel
  - Can safely walk pointer dereferencing among structures.

What’s new? (to you!)

- Kprobe trace (2009)
  - A lot of people still do not know about this
  - Dynamically create trace events almost anywhere in the kernel
  - Can safely walk pointer dereferencing among structures.
  - Can easily reference arguments (i.e. “`$arg1`”) (2018)

What’s new? (to you!)

- Kprobe trace (2009)
  - A lot of people still do not know about this
  - Dynamically create trace events almost anywhere in the kernel
  - Can safely walk pointer dereferencing among structures.
  - Can easily reference arguments (i.e. “`$arg1`”) (2018)
  - A little complex, so people don’t always use them

Documentation/trace/kprobetrace.rst

```
Synopsis of kprobe_events
-------------------------
::

  p[:[GRP/]EVENT] [MOD:]SYM[+offs]|MEMADDR [FETCHARGS]  : Set a probe
  r[MAXACTIVE][:[GRP/]EVENT] [MOD:]SYM[+0] [FETCHARGS]  : Set a return probe
  p:[GRP/]EVENT] [MOD:]SYM[+0]%return [FETCHARGS]       : Set a return probe
  -:[GRP/]EVENT                                         : Clear a probe

 GRP            : Group name. If omitted, use "kprobes" for it.
 EVENT          : Event name. If omitted, the event name is generated
                  based on SYM+offs or MEMADDR.
 MOD            : Module name which has given SYM.
 SYM[+offs]     : Symbol+offset where the probe is inserted.
 SYM%return     : Return address of the symbol
 MEMADDR        : Address where the probe is inserted.
 MAXACTIVE      : Maximum number of instances of the specified function that
                  can be probed simultaneously, or 0 for the default value
                  as defined in Documentation/trace/kprobes.rst section 1.3.1.

 FETCHARGS      : Arguments. Each probe can have up to 128 args.
  %REG          : Fetch register REG
  @ADDR         : Fetch memory at ADDR (ADDR should be in kernel)
  @SYM[+|-offs] : Fetch memory at SYM +|- offs (SYM should be a data symbol)
  $stackN       : Fetch Nth entry of stack (N >= 0)
  $stack        : Fetch stack address.
  $argN         : Fetch the Nth function argument. (N >= 1) (\*1)
  $retval       : Fetch return value.(\*2)
  $comm         : Fetch current task comm.
  +|-[u]OFFS(FETCHARG) : Fetch memory at FETCHARG +|- OFFS address.(\*3)(\*4)
  \IMM          : Store an immediate value to the argument.
  NAME=FETCHARG : Set NAME as the argument name of FETCHARG.
  FETCHARG:TYPE : Set TYPE as the type of FETCHARG. Currently, basic types
                  (u8/u16/u32/u64/s8/s16/s32/s64), hexadecimal types
                  (x8/x16/x32/x64), "string", "ustring" and bitfield
                  are supported.
```

Example kprobe trace

```
# trace-cmd list -f ip_rcv
ip_rcv_finish_core.constprop.0
ip_rcv_core
ip_rcv_finish
ip_rcv
```

Example kprobe trace

```
# trace-cmd start -p function -l 'ip_rcv*'
# trace-cmd show
# tracer: function
#
# entries-in-buffer/entries-written: 246/246   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] ..s2. 66116.392123: ip_rcv_core <-ip_list_rcv
          <idle>-0       [001] ..s2. 66116.392124: ip_rcv_finish_core.constprop.0 <-ip_sublist_rcv
          <idle>-0       [001] ..s2. 66117.343512: ip_rcv_core <-ip_list_rcv
          <idle>-0       [001] ..s2. 66117.343515: ip_rcv_finish_core.constprop.0 <-ip_sublist_rcv
          <idle>-0       [001] ..s2. 66117.632545: ip_rcv_core <-ip_list_rcv
```

Example kprobe trace

```
# trace-cmd start -p function -l 'ip_rcv*'
# trace-cmd show
# tracer: function
#
# entries-in-buffer/entries-written: 246/246   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] ..s2. 66116.392123: ip_rcv_core <-ip_list_rcv
          <idle>-0       [001] ..s2. 66116.392124: ip_rcv_finish_core.constprop.0 <-ip_sublist_rcv
          <idle>-0       [001] ..s2. 66117.343512: ip_rcv_core <-ip_list_rcv
          <idle>-0       [001] ..s2. 66117.343515: ip_rcv_finish_core.constprop.0 <-ip_sublist_rcv
          <idle>-0       [001] ..s2. 66117.632545: ip_rcv_core <-ip_list_rcv
```

net/ipv4/ip_input.c

```
static struct sk_buff *ip_rcv_core(struct sk_buff *skb, struct net *net)
{
      const struct iphdr *iph;
      int drop_reason;
      u32 len;

      /* When the interface is in promisc. mode, drop all the crap
       * that it receives, do not try to analyse it.
       */
      if (skb->pkt_type == PACKET_OTHERHOST) {
            drop_reason = SKB_DROP_REASON_OTHERHOST;
            goto drop;
      }

      __IP_UPD_PO_STATS(net, IPSTATS_MIB_IN, skb->len);

      skb = skb_share_check(skb, GFP_ATOMIC);
      if (!skb) {
            __IP_INC_STATS(net, IPSTATS_MIB_INDISCARDS);
            goto out;
      }
```

Example kprobe trace

```
# trace-cmd reset
# echo 'p:ip_rcv ip_rcv_core skb=$arg1 net=$arg2' > /sys/kernel/tracing/kprobe_events
# trace-cmd list -e kprobes -F --full

system: kprobes
name: ip_rcv
ID: 1794
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 skb;   offset:16;      size:8; signed:0;
        field:u64 net;   offset:24;      size:8; signed:0;

print fmt: "(%lx) skb=0x%Lx net=0x%Lx", REC->__probe_ip, REC->skb, REC->net
```

Example kprobe trace

```
# trace-cmd start -e ip_rcv
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 262/262   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] ..s1. 66567.387728: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023800 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.387799: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023b00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.730430: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407d46d00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.867413: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023600 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.869317: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023400 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.943534: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407d46c00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.037256: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023c00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.038921: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023700 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.118077: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023f00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.119799: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023e00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.219771: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023300 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.220618: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023b00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.316214: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023600 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.317974: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023400 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66570.951077: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023f00 net=0xffffffff84064a40
            bash-1301    [001] ..s.. 66571.029691: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023e00 net=0xffffffff84064a40
```

Example kprobe trace

```
# trace-cmd start -e ip_rcv
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 262/262   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] ..s1. 66567.387128: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023800 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.387799: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023b00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.730430: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407d46d00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.867413: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023600 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.869317: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023400 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66567.943534: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407d46c00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.037256: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023c00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.038921: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023700 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.118077: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023f00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.119799: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023e00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.219771: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023300 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.220618: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023b00 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.316214: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023600 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66568.317974: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023400 net=0xffffffff84064a40
          <idle>-0       [001] ..s1. 66570.951077: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023f00 net=0xffffffff84064a40
            bash-1301    [001] ..s.. 66571.029691: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e403023e00 net=0xffffffff84064a40
```

BORING!!!

Example kprobe trace

```
skb=0xffff92e403023800 net=0xffffffff84064a40
skb=0xffff92e403023b00 net=0xffffffff84064a40
skb=0xffff92e407d46d00 net=0xffffffff84064a40
skb=0xffff92e403023600 net=0xffffffff84064a40
skb=0xffff92e403023400 net=0xffffffff84064a40
skb=0xffff92e407d46c00 net=0xffffffff84064a40
skb=0xffff92e403023c00 net=0xffffffff84064a40
skb=0xffff92e403023700 net=0xffffffff84064a40
skb=0xffff92e403023f00 net=0xffffffff84064a40
skb=0xffff92e403023e00 net=0xffffffff84064a40
skb=0xffff92e403023300 net=0xffffffff84064a40
skb=0xffff92e403023b00 net=0xffffffff84064a40
skb=0xffff92e403023600 net=0xffffffff84064a40
skb=0xffff92e403023400 net=0xffffffff84064a40
skb=0xffff92e403023f00 net=0xffffffff84064a40
skb=0xffff92e403023e00 net=0xffffffff84064a40
```

Example kprobe trace

```
skb=0xffff92e403023800 net=0xffffffff84064a40
skb=0xffff92e403023b00 net=0xffffffff84064a40
skb=0xffff92e407d46d00 net=0xffffffff84064a40
skb=0xffff92e403023600 net=0xffffffff84064a40
skb=0xffff92e403023400 net=0xffffffff84064a40
skb=0xffff92e407d46c00 net=0xffffffff84064a40
skb=0xffff92e403023c00 net=0xffffffff84064a40
skb=0xffff92e403023700 net=0xffffffff84064a40
skb=0xffff92e403023f00 net=0xffffffff84064a40
skb=0xffff92e403023e00 net=0xffffffff84064a40
skb=0xffff92e403023300 net=0xffffffff84064a40
skb=0xffff92e403023b00 net=0xffffffff84064a40
skb=0xffff92e403023600 net=0xffffffff84064a40
skb=0xffff92e403023400 net=0xffffffff84064a40
skb=0xffff92e403023f00 net=0xffffffff84064a40
skb=0xffff92e403023e00 net=0xffffffff84064a40
```

MOSTLY USELESS!

net/ipv4/ip_input.c

```
static struct sk_buff *ip_rcv_core(struct sk_buff *skb, struct net *net)
{
      const struct iphdr *iph;
      int drop_reason;
      u32 len;

      /* When the interface is in promisc. mode, drop all the crap
       * that it receives, do not try to analyse it.
       */
      if (skb->pkt_type == PACKET_OTHERHOST) {
            drop_reason = SKB_DROP_REASON_OTHERHOST;
            goto drop;
      }

      __IP_UPD_PO_STATS(net, IPSTATS_MIB_IN, skb->len);

      skb = skb_share_check(skb, GFP_ATOMIC);
      if (!skb) {
            __IP_INC_STATS(net, IPSTATS_MIB_INDISCARDS);
            goto out;
      }
```

include/linux/skbuff.h

```
struct sk_buff {
      union {
            struct {
                  /* These two members must be first to match sk_buff_head. */
                  struct sk_buff          *next;
                  struct sk_buff          *prev;

                  union {
                        struct net_device *dev;
                        /* Some protocols might use this space to store information,
                         * while device pointer would be NULL.
                         * UDP receive path is one user.
                         */
                        unsigned long           dev_scratch;
                  };
            };
            struct rb_node          rbnode; /* used in netem, ip4 defrag, and tcp stack
*/
            struct list_head  list;
            struct llist_node ll_node;
      };
```

include/linux/netdevice.h

```
struct net_device {
      char              name[IFNAMSIZ];
      struct netdev_name_node *name_node;
      struct dev_ifalias__rcu *ifalias;
      /*
       *    I/O specific fields
       *    FIXME: Merge these and struct ifmap into one
       */
      unsigned long           mem_end;
      unsigned long           mem_start;
      unsigned long           base_addr;
```

Use gdb on the vmlinux kernel

```
$ gdb vmlinux

(gdb) p &((struct sk_buff *)0)->dev
$12 = (struct net_device **) 0x10 <fixed_percpu_data+16>
```

Use gdb on the vmlinux kernel

```
$ gdb vmlinux

(gdb) p &((struct sk_buff *)0)->dev
$12 = (struct net_device **) 0x10 <fixed_percpu_data+16>

(gdb) p &((struct net_device *)0)->name
$13 = (char (*)[16]) 0x0 <fixed_percpu_data>
```

Example kprobe trace

```
# trace-cmd reset
# echo 'p:ip_rcv ip_rcv_core skb=$arg1 dev=+0(+0x10($arg1))' > /sys/kernel/tracing/kprobe_events
# trace-cmd list -e kprobes -F --full

system: kprobes
name: ip_rcv
ID: 1795
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 skb;   offset:16;      size:8; signed:0;
        field:u64 dev;   offset:24;      size:8; signed:0;

print fmt: "(%lx) skb=0x%Lx dev=0x%Lx", REC->__probe_ip, REC->skb, REC->dev
```

Example kprobe trace

```
# trace-cmd reset
# echo 'p:ip_rcv ip_rcv_core skb=$arg1 dev=+0(+0x10($arg1))' > /sys/kernel/tracing/kprobe_events
# trace-cmd list -e kprobes -F --full

system: kprobes
name: ip_rcv
ID: 1795
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 skb;   offset:16;      size:8; signed:0;
        Field:u64 dev;   offset:24;      size:8; signed:0;

print fmt: "(%lx) skb=0x%Lx dev=0x%Lx", REC->__probe_ip, REC->skb, REC->dev
```

Example kprobe trace

```
# trace-cmd reset
# echo 'p:ip_rcv ip_rcv_core skb=$arg1 dev=+0(+0x10($arg1))' > /sys/kernel/tracing/kprobe_events
# trace-cmd list -e kprobes -F --full

system: kprobes
name: ip_rcv
ID: 1795
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 skb;   offset:16;      size:8; signed:0;
        field:u64 dev;   offset:24;      size:8; signed:0;

print fmt: "(%lx) skb=0x%Lx dev=0x%Lx", REC->__probe_ip, REC->skb, REC->dev
```

Example kprobe trace

```
# trace-cmd start -e ip_rcv
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 15/15   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] ..s1. 68209.664309: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e41fc29200 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68209.664392: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56300 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.000780: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56700 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.002727: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56b00 dev=0x307331706e65
          <idle>-0       [001] .Ns1. 68210.090940: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e41fc29200 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.138308: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56600 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.138841: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56800 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.196663: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56500 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.198163: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56e00 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.271374: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56000 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.272137: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56400 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.396426: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56900 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.398085: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56700 dev=0x307331706e65
          <idle>-0       [001] .Ns1. 68211.258729: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56600 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68211.260671: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56800 dev=0x307331706e65
```

Example kprobe trace

```
# trace-cmd start -e ip_rcv
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 15/15   #P:2
#
#                              _-----=> irqs-off/BH-disabled
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| / _-=> migrate-disable
#                            |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] ..s1. 68209.664309: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e41fc29200 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68209.664392: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56300 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.000700: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56700 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.002720: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56b00 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.090940: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e41fc29200 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.138803: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56600 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.138841: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56800 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.196663: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56500 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.198163: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56e00 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.271374: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56000 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.272137: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56400 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.396426: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56900 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68210.398085: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56700 dev=0x307331706e65
          <idle>-0       [001] .Ns1. 68211.258729: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56600 dev=0x307331706e65
          <idle>-0       [001] ..s1. 68211.260671: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414c56800 dev=0x307331706e65
```

STILL BORING!!!

Documentation/trace/kprobetrace.rst

```
Synopsis of kprobe_events
-------------------------
::

  p[:[GRP/]EVENT] [MOD:]SYM[+offs]|MEMADDR [FETCHARGS]  : Set a probe
  r[MAXACTIVE][:[GRP/]EVENT] [MOD:]SYM[+0] [FETCHARGS]  : Set a return probe
  p:[GRP/]EVENT] [MOD:]SYM[+0]%return [FETCHARGS]       : Set a return probe
  -:[GRP/]EVENT                                         : Clear a probe

 GRP            : Group name. If omitted, use "kprobes" for it.
 EVENT          : Event name. If omitted, the event name is generated
                  based on SYM+offs or MEMADDR.
 MOD            : Module name which has given SYM.
 SYM[+offs]     : Symbol+offset where the probe is inserted.
 SYM%return     : Return address of the symbol
 MEMADDR        : Address where the probe is inserted.
 MAXACTIVE      : Maximum number of instances of the specified function that
                  can be probed simultaneously, or 0 for the default value
                  as defined in Documentation/trace/kprobes.rst section 1.3.1.

 FETCHARGS      : Arguments. Each probe can have up to 128 args.
  %REG          : Fetch register REG
  @ADDR         : Fetch memory at ADDR (ADDR should be in kernel)
  @SYM[+|-offs] : Fetch memory at SYM +|- offs (SYM should be a data symbol)
  $stackN       : Fetch Nth entry of stack (N >= 0)
  $stack        : Fetch stack address.
  $argN         : Fetch the Nth function argument. (N >= 1) (\*1)
  $retval       : Fetch return value.(\*2)
  $comm         : Fetch current task comm.
  +|-[u]OFFS(FETCHARG) : Fetch memory at FETCHARG +|- OFFS address.(\*3)(\*4)
  \IMM          : Store an immediate value to the argument.
  NAME=FETCHARG : Set NAME as the argument name of FETCHARG.
  FETCHARG:TYPE : Set TYPE as the type of FETCHARG. Currently, basic types
                  (u8/u16/u32/u64/s8/s16/s32/s64), hexadecimal types
                  (x8/x16/x32/x64), "string", "ustring" and bitfield
                  are supported.
```

Documentation/trace/kprobetrace.rst

```
Synopsis of kprobe_events
-------------------------
::

  p[:[GRP/]EVENT] [MOD:]SYM[+offs]|MEMADDR [FETCHARGS]  : Set a probe
  r[MAXACTIVE][:[GRP/]EVENT] [MOD:]SYM[+0] [FETCHARGS]  : Set a return probe
  p:[GRP/]EVENT] [MOD:]SYM[+0]%return [FETCHARGS]       : Set a return probe
  -:[GRP/]EVENT                                         : Clear a probe

 GRP            : Group name. If omitted, use "kprobes" for it.
 EVENT          : Event name. If omitted, the event name is generated
                  based on SYM+offs or MEMADDR.
 MOD            : Module name which has given SYM.
 SYM[+offs]     : Symbol+offset where the probe is inserted.
 SYM%return     : Return address of the symbol
 MEMADDR        : Address where the probe is inserted.
 MAXACTIVE      : Maximum number of instances of the specified function that
                  can be probed simultaneously, or 0 for the default value
                  as defined in Documentation/trace/kprobes.rst section 1.3.1.

 FETCHARGS      : Arguments. Each probe can have up to 128 args.
  %REG          : Fetch register REG
  @ADDR         : Fetch memory at ADDR (ADDR should be in kernel)
  @SYM[+|-offs] : Fetch memory at SYM +|- offs (SYM should be a data symbol)
  $stackN       : Fetch Nth entry of stack (N >= 0)
  $stack        : Fetch stack address.
  $argN         : Fetch the Nth function argument. (N >= 1) (\*1)
  $retval       : Fetch return value.(\*2)
  $comm         : Fetch current task comm.
  +|-[u]OFFS(FETCHARG) : Fetch memory at FETCHARG +|- OFFS address.(\*3)(\*4)
  \IMM          : Store an immediate value to the argument.
  NAME=FETCHARG : Set NAME as the argument name of FETCHARG.
  FETCHARG:TYPE : Set TYPE as the type of FETCHARG. Currently, basic types
                  (u8/u16/u32/u64/s8/s16/s32/s64), hexadecimal types
                  (x8/x16/x32/x64), "string", "ustring" and bitfield
                  are supported.
```

Example kprobe trace

```
# trace-cmd reset
# echo 'p:ip_rcv ip_rcv_core skb=$arg1 dev=+0(+0x10($arg1)):string' > /sys/kernel/tracing/kprobe_events
# trace-cmd list -e kprobes -F --full

system: kprobes
name: ip_rcv
ID: 1795
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 skb;   offset:16;      size:8; signed:0;
        field:__data_loc char[] dev;    offset:24;      size:4; signed:1;

print fmt: "(%lx) skb=0x%Lx dev=\"%s\"", REC->__probe_ip, REC->skb, __get_str(dev)
```

Example kprobe trace

```
# trace-cmd start -e ip_rcv
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 14/14   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] ..s1. 68524.281334: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414eb2f00 dev="enp1s0"
          <idle>-0       [001] ..s1. 68524.901629: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e4042dbc00 dev="enp1s0"
          <idle>-0       [001] ..s1. 68525.251421: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edcf00 dev="enp1s0"
          <idle>-0       [001] ..s1. 68525.252026: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc500 dev="enp1s0"
          <idle>-0       [001] ..s1. 68525.330692: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e4042dbc00 dev="enp1s0"
          <idle>-0       [001] ..s1. 68525.407229: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edce00 dev="enp1s0"
          <idle>-0       [001] ..s1. 68525.407707: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc000 dev="enp1s0"
          <idle>-0       [001] ..s1. 68525.470059: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc900 dev="enp1s0"
          <idle>-0       [001] ..s1. 68525.470552: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc800 dev="enp1s0"
          <idle>-0       [001] .Ns1. 68525.863100: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edcc00 dev="enp1s0"
          <idle>-0       [001] ..s1. 68526.335182: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc300 dev="enp1s0"
          <idle>-0       [001] .Ns1. 68527.065694: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc100 dev="enp1s0"
          <idle>-0       [001] ..s1. 68527.066994: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc200 dev="enp1s0"
          <idle>-0       [001] ..s1. 68527.067117: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edcf00 dev="enp1s0"
```

Example kprobe trace

```
# trace-cmd start -e ip_rcv
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 14/14   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] ..s1.  68524.2[illegible]334: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e414eb2f00 dev="enp1s0"
          <idle>-0       [001] ..s1.  68525.[illegible]: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e4042dbc00 dev="enp1s0"
          <idle>-0       [001] ..s1.  68525.251421: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edcf00 dev="enp1s0"
          <idle>-0       [001] ..s1.  68525.252026: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc500 dev="enp1s0"
          <idle>-0       [001] ..s1.  68525.330692: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e4042dbc00 dev="enp1s0"
          <idle>-0       [001] ..s1.  68525.407229: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edce00 dev="enp1s0"
          <idle>-0       [001] ..s1.  68525.407707: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc000 dev="enp1s0"
          <idle>-0       [001] ..s1.  68525.470059: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc900 dev="enp1s0"
          <idle>-0       [001] ..s1.  68525.470552: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc800 dev="enp1s0"
          <idle>-0       [001] .Ns1.  68525.863100: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edcc00 dev="enp1s0"
          <idle>-0       [001] ..s1.  68526.335182: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc300 dev="enp1s0"
          <idle>-0       [001] .Ns1.  68527.065694: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc100 dev="enp1s0"
          <idle>-0       [001] ..s1.  68527.066994: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edc200 dev="enp1s0"
          <idle>-0       [001] ..s1.  68527.067117: ip_rcv: (ip_rcv_core+0x0/0x350) skb=0xffff92e407edcf00 dev="enp1s0"
```

EXCITING!!!

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
  - Just like kprobe tracing but for user space

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
  - Just like kprobe tracing but for user space
  - Triggered via breakpoints

What’s new? (to you!)

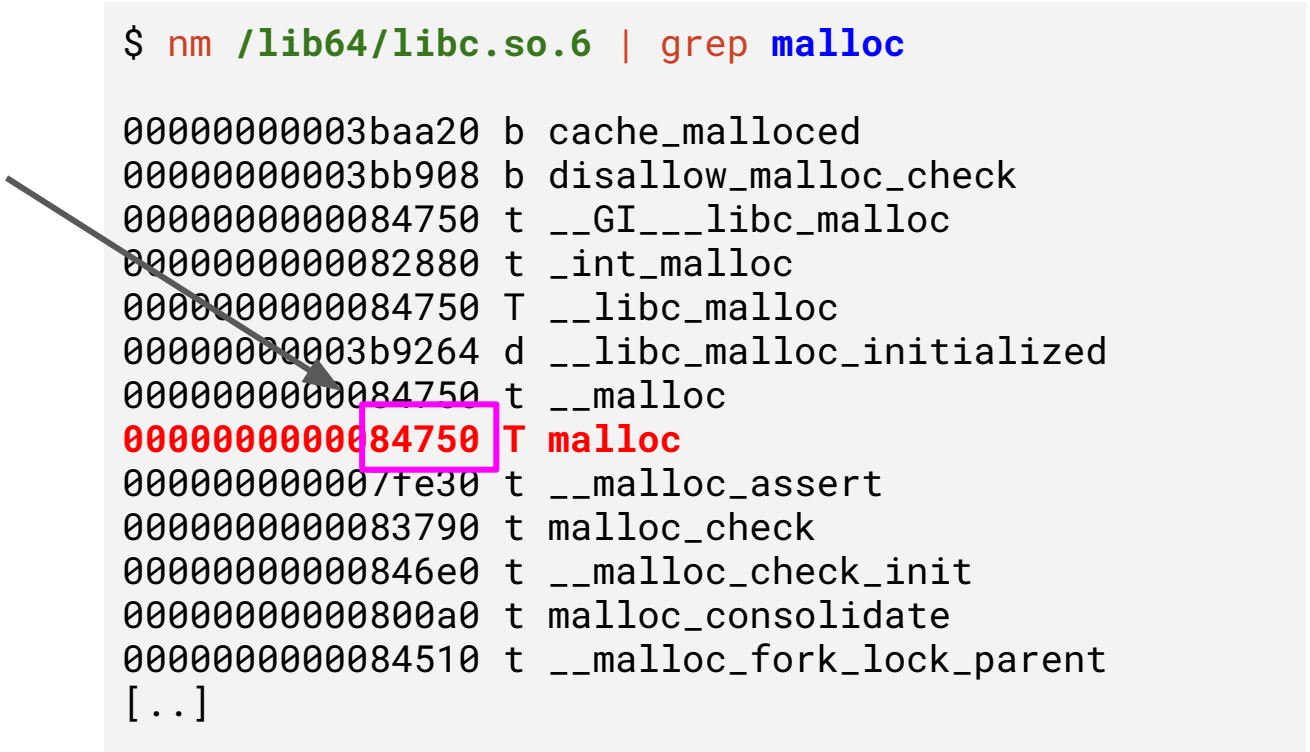
- Kprobe trace (2009)
- Uprobe trace (2012)
  - Just like kprobe tracing but for user space
  - Triggered via breakpoints
  - Documented in Documentation/trace/uprobetracer.rst

Finding malloc

```
$ nm /lib64/libc.so.6 | grep malloc

00000000003baa20 b cache_malloced
00000000003bb908 b disallow_malloc_check
0000000000084750 t __GI___libc_malloc
0000000000082880 t _int_malloc
0000000000084750 T __libc_malloc
00000000003b9264 d __libc_malloc_initialized
0000000000084750 t __malloc
0000000000084750 T malloc
000000000007fe30 t __malloc_assert
0000000000083790 t malloc_check
00000000000846e0 t __malloc_check_init
00000000000800a0 t malloc_consolidate
0000000000084510 t __malloc_fork_lock_parent
[..]
```

Finding malloc



Example uprobe trace

```
# trace-cmd reset
# echo 'p:malloc /lib64/libc.so.6:0x84750 size=%di:u64' > /sys/kernel/tracing/uprobe_events
# trace-cmd list -e uprobes:malloc -F --full

system: uprobes
name: malloc
ID: 1800
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 size; offset:16;      size:8; signed:0;

print fmt: "(%lx) size=0x%Lx", REC->__probe_ip, REC->size
```

Example uprobe trace

```
# trace-cmd reset
# echo 'p:malloc /lib64/libc.so.6:0x84750 size=%di:u64' > /sys/kernel/tracing/uprobe_events
# trace-cmd list -e uprobes:malloc -F --full

system: uprobes
name: malloc
ID: 1800
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 size; offset:16;      size:8; signed:0;

print fmt: "(%lx) size=0x%Lx", REC->__probe_ip, REC->size
```

Example uprobe trace

```
# trace-cmd start -e malloc
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 1819/1819   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
            bash-1301    [000] DNZff 75588.599156: malloc: (0x7f02df284750) size=3
            bash-1301    [000] DNZff 75588.599162: malloc: (0x7f02df284750) size=4
            bash-1301    [000] DNZff 75588.599168: malloc: (0x7f02df284750) size=2
            bash-1301    [000] DNZff 75588.599195: malloc: (0x7f02df284750) size=16
            bash-1301    [000] DNZff 75588.599196: malloc: (0x7f02df284750) size=32
            bash-1301    [000] DNZff 75588.599197: malloc: (0x7f02df284750) size=32
            bash-1301    [000] DNZff 75588.599198: malloc: (0x7f02df284750) size=32
            bash-1301    [000] DNZff 75588.599198: malloc: (0x7f02df284750) size=2
            bash-1301    [000] DNZff 75588.599200: malloc: (0x7f02df284750) size=7
            bash-1301    [000] DNZff 75588.599200: malloc: (0x7f02df284750) size=76
            bash-1301    [000] DNZff 75588.599202: malloc: (0x7f02df284750) size=32
            bash-1301    [000] DNZff 75588.599202: malloc: (0x7f02df284750) size=228
            bash-1301    [000] DNZff 75588.599203: malloc: (0x7f02df284750) size=32
            bash-1301    [000] DNZff 75588.599204: malloc: (0x7f02df284750) size=32
            bash-1301    [000] DNZff 75588.599204: malloc: (0x7f02df284750) size=32
```

Example uprobe trace

```
# echo 'r:malloc /lib64/libc.so.6:0x84750 ret=%ax' >> /sys/kernel/tracing/uprobe_events
# trace-cmd list -e uprobes:malloc -F --full

system: uprobes
name: malloc
ID: 1800
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 size; offset:16;      size:8; signed:0;

print fmt: "(%lx) size=0x%Lx", REC->__probe_ip, REC->size
```

Example uprobe trace

```
# echo 'r:malloc /lib64/libc.so.6:0x84750 ret=%ax' >> /sys/kernel/tracing/uprobe_events
# trace-cmd list -e uprobes:malloc -F --full

system: uprobes
name: malloc
ID: 1800
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 size; offset:16;      size:8; signed:0;

print fmt: "(%lx) size=0x%Lx", REC->__probe_ip, REC->size
```

Example uprobe trace

```
# trace-cmd start -e uprobes
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 1464/1464   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
            bash-1301    [001] DNZff 127208.228570: malloc: (0x7f02df284750) size=3
            bash-1301    [001] DNZff 127208.228611: malloc_ret: (0x55b9e6089442 <- 0x7f02df284750) ret5=0x55b9e689fa90
            bash-1301    [001] DNZff 127208.228632: malloc: (0x7f02df284750) size=4
            bash-1301    [001] DNZff 127208.228636: malloc_ret: (0x55b9e6089442 <- 0x7f02df284750) ret5=0x55b9e6848c70
            bash-1301    [001] DNZff 127208.228667: malloc: (0x7f02df284750) size=2
            bash-1301    [001] DNZff 127208.228671: malloc_ret: (0x55b9e6089442 <- 0x7f02df284750) ret5=0x55b9e68cc360
            bash-1301    [001] DNZff 127208.228784: malloc: (0x7f02df284750) size=16
            bash-1301    [001] DNZff 127208.228789: malloc_ret: (0x55b9e6089442 <- 0x7f02df284750) ret5=0x55b9e688b410
            bash-1301    [001] DNZff 127208.228793: malloc: (0x7f02df284750) size=32
            bash-1301    [001] DNZff 127208.228798: malloc_ret: (0x55b9e6089442 <- 0x7f02df284750) ret5=0x55b9e689b710
            bash-1301    [001] DNZff 127208.228801: malloc: (0x7f02df284750) size=32
            bash-1301    [001] DNZff 127208.228805: malloc_ret: (0x55b9e6089442 <- 0x7f02df284750) ret5=0x55b9e689d540
            bash-1301    [001] DNZff 127208.228808: malloc: (0x7f02df284750) size=32
            bash-1301    [001] DNZff 127208.228812: malloc_ret: (0x55b9e6089442 <- 0x7f02df284750) ret5=0x55b9e67fb900
            bash-1301    [001] DNZff 127208.228820: malloc: (0x7f02df284750) size=2
```

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
  - An event “trigger”

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
  - An event “trigger”
  - Can do counting of event fields

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
  - An event “trigger”
  - Can do counting of event fields
  - Documented in Documentation/trace/histogram.rst

Example histogram

```
# trace-cmd list -e raw_syscall:sys_enter -F --full

system: raw_syscalls
name: sys_enter
ID: 338
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:long id;  offset:8;       size:8; signed:1;
        field:unsigned long args[6];    offset:16;      size:48;        signed:0;
```

Example histogram

```
# cd /sys/kernel/tracing
# echo 'hist:keys=id' > events/raw_syscall/sys_enter/trigger
```

Example histogram

```
# cd /sys/kernel/tracing
# echo 'hist:keys=id' > events/raw_syscall/sys_enter/trigger
# cat events/raw_syscall/sys_enter/hist
# event histogram
#
# trigger info: hist:keys=id:vals=hitcount:sort=hitcount:size=2048 [active]
#

{ id:         11 } hitcount:          1
{ id:         87 } hitcount:          1
{ id:         59 } hitcount:          1
{ id:         21 } hitcount:          1
{ id:        158 } hitcount:          1
{ id:        221 } hitcount:          1
{ id:          7 } hitcount:          2
[..]
{ id:         72 } hitcount:         31
{ id:        257 } hitcount:         38
{ id:          1 } hitcount:         42
{ id:          0 } hitcount:         47
{ id:          3 } hitcount:         51
{ id:         23 } hitcount:         54
{ id:         13 } hitcount:         63
{ id:        228 } hitcount:         98
{ id:         14 } hitcount:        138

Totals:
    Hits: 741
    Entries: 34
    Dropped: 0
```

Example histogram

```
# cd /sys/kernel/tracing
# echo 'hist:keys=id' > events/raw_syscall/sys_enter/trigger
# cat events/raw_syscall/sys_enter/hist
# event histogram
#
# trigger info: hist:keys=id:vals=hitcount:sort=hitcount:size=2048 [active]
#

{ id:         11 } hitcount:          1
{ id:         87 } hitcount:          1
{ id:         59 } hitcount:          1
{ id:         21 } hitcount:          1
{ id:        158 } hitcount:          1
{ id:        221 } hitcount:          1
{ id:          7 } hitcount:          2
[..]
{ id:          [illegible] } hitcount:          3
{ id:          [illegible] } hitcount:          [illegible]
{ id:          [illegible] } hitcount:         42
{ id:          [illegible] } hitcount:         47
{ id:          3 } hitcount:         51
{ id:         23 } hitcount:         54
{ id:         13 } hitcount:         63
{ id:        228 } hitcount:         98
{ id:         14 } hitcount:        138

Totals:
    Hits: 741
    Entries: 34
    Dropped: 0
```

NOT HELPFUL

Documentation/trace/histogram.rst

```
        =============  ================================================
        .hex           display a number as a hex value
        .sym           display an address as a symbol
        .sym-offset    display an address as a symbol and offset
        .syscall       display a syscall id as a system call name
        .execname      display a common_pid as a program name
        .log2          display log2 value rather than raw number
        .buckets=size  display grouping of values rather than raw number
        .usecs         display a common_timestamp in microseconds
        =============  ================================================
```

Documentation/trace/histogram.rst

```
        =============  ================================================
        .hex           display a number as a hex value
        .sym           display an address as a symbol
        .sym-offset    display an address as a symbol and offset
        .syscall       display a syscall id as a system call name
        .execname      display a common_pid as a program name
        .log2          display log2 value rather than raw number
        .buckets=size  display grouping of values rather than raw number
        .usecs         display a common_timestamp in microseconds
        =============  ================================================
```

Example histogram

```
# cd /sys/kernel/tracing
# echo 'hist:keys=id.syscall' > events/raw_syscall/sys_enter/trigger
```

Example histogram

```
# cd /sys/kernel/tracing
# echo 'hist:keys=id.syscall' > events/raw_syscall/sys_enter/trigger
# cat events/raw_syscall/sys_enter/hist
# event histogram
#
# trigger info: hist:keys=id.syscall:vals=hitcount:sort=hitcount:size=2048 [active]
#

{ id: sys_newstat                  [  4] } hitcount:          1
{ id: sys_fadvise64                [221] } hitcount:          1
{ id: sys_arch_prctl               [158] } hitcount:          1
{ id: sys_inotify_add_watch        [254] } hitcount:          1
{ id: sys_pipe                     [ 22] } hitcount:          1
{ id: sys_munmap                   [ 11] } hitcount:          1
{ id: sys_wait4                    [ 61] } hitcount:          1
{ id: sys_execve                   [ 59] } hitcount:          1
[...]
{ id: sys_openat                   [257] } hitcount:         31
{ id: sys_ioctl                    [ 16] } hitcount:         31
{ id: sys_read                     [  0] } hitcount:         43
{ id: sys_write                    [  1] } hitcount:         45
{ id: sys_select                   [ 23] } hitcount:         58
{ id: sys_rt_sigaction             [ 13] } hitcount:         81
{ id: sys_rt_sigprocmask           [ 14] } hitcount:        143
{ id: sys_clock_gettime            [228] } hitcount:        150

Totals:
    Hits: 737
    Entries: 32
    Dropped: 0
```

Example histogram

```
# cd /sys/kernel/tracing
# echo 'hist:keys=id.syscall' > events/raw_syscall/sys_enter/trigger
# cat events/raw_syscall/sys_enter/hist
# event histogram
#
# trigger info: hist:keys=id.syscall:vals=hitcount:sort=hitcount:size=2048 [active]
#

{ id: sys_newstat                [  4] } hitcount:          1
{ id: sys_fadvise64              [221] } hitcount:          1
{ id: sys_arch_prctl             [158] } hitcount:          1
{ id: sys_inotify_add_watch      [254] } hitcount:          1
{ id: sys_pipe                   [ 22] } hitcount:          2
{ id: sys_munmap                 [ 11] } hitcount:          2
{ id: sys_wait4                  [ 61] } hitcount:          3
{ id: sys_execve                 [ 59] } hitcount:          4
[...]
{ id: sys_openat                 [257] } hitcount:         31
{ id: sys_ioctl                  [ 16] } hitcount:         31
{ id: sys_read                   [  0] } hitcount:         43
{ id: sys_write                  [  1] } hitcount:         45
{ id: sys_select                 [ 23] } hitcount:         58
{ id: sys_rt_sigaction           [ 13] } hitcount:         81
{ id: sys_rt_sigprocmask         [ 14] } hitcount:        143
{ id: sys_clock_gettime          [228] } hitcount:        150

Totals:
    Hits: 737
    Entries: 32
    Dropped: 0
```

HELPFUL

Example histogram on uprobe

```
# trace-cmd reset
# echo 'p:malloc /lib64/libc.so.6:0x84750 size=%di:u64' > /sys/kernel/tracing/uprobe_events
# trace-cmd list -e uprobes:malloc -F --full

system: uprobes
name: malloc
ID: 1800
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:unsigned long __probe_ip; offset:8;       size:8; signed:0;
        field:u64 size; offset:16;      size:8; signed:0;

print fmt: "(%lx) size=0x%Lx", REC->__probe_ip, REC->size
```

Example histogram on uprobe

```
# cd /sys/kernel/tracing
# echo 'hist:keys=common_pid:vals=size:sort=size' > events/uprobes/malloc/trigger
```

Example histogram on uprobe

```
# cd /sys/kernel/tracing
# echo 'hist:keys=common_pid:vals=size:sort=size' > events/uprobes/malloc/trigger
# cat events/uprobes/malloc/hist
# event histogram
#
# trigger info: hist:keys=common_pid:vals=hitcount,size:sort=size:size=2048 [active]
#

{ common_pid:        984 } hitcount:          2  size:         30
{ common_pid:       1300 } hitcount:          2  size:        768
{ common_pid:          1 } hitcount:         24  size:       6607
{ common_pid:       1301 } hitcount:        522  size:      21831
{ common_pid:       1140 } hitcount:          8  size:      45824
{ common_pid:        558 } hitcount:         74  size:      97327
{ common_pid:       3755 } hitcount:        418  size:     161921
{ common_pid:       3754 } hitcount:        418  size:     161921

Totals:
    Hits: 1468
    Entries: 8
    Dropped: 0
```

Example histogram on uprobe

```
# cd /sys/kernel/tracing
# echo 'hist:keys=common_pid:vals=size:sort=size' > events/uprobes/malloc/trigger
# cat events/uprobes/malloc/hist
# event histogram
#
# trigger info: hist:keys=common_pid:vals=hitcount,size:sort=size:size=2048 [active]
#

{ common_pid:       984 } hitcount:          1  size:            4
{ common_pid:      1300 } hitcount:          2  size:          768
{ common_pid:         1 } hitcount:         24  size:         6607
{ common_pid:      1301 } hitcount:        522  size:        21831
{ common_pid:      1140 } hitcount:          8  size:        45824
{ common_pid:       558 } hitcount:         74  size:        97327
{ common_pid:      3755 } hitcount:         41  size:       167936
{ common_pid:      3754 } hitcount:         41  size:       167936

Totals:
    Hits: 1468
    Entries: 8
    Dropped: 0
```

SORTA USEFUL

Documentation/trace/histogram.rst

```
        =============  ================================================
        .hex           display a number as a hex value
        .sym           display an address as a symbol
        .sym-offset    display an address as a symbol and offset
        .syscall       display a syscall id as a system call name
        .execname      display a common_pid as a program name
        .log2          display log2 value rather than raw number
        .buckets=size  display grouping of values rather than raw number
        .usecs         display a common_timestamp in microseconds
        =============  ================================================
```

Documentation/trace/histogram.rst

```
        =============  ================================================
        .hex           display a number as a hex value
        .sym           display an address as a symbol
        .sym-offset    display an address as a symbol and offset
        .syscall       display a syscall id as a system call name
        .execname      display a common_pid as a program name
        .log2          display log2 value rather than raw number
        .buckets=size  display grouping of values rather than raw number
        .usecs         display a common_timestamp in microseconds
        =============  ================================================
```

Example histogram on uprobe

```
# cd /sys/kernel/tracing
# echo 'hist:keys=common_pid.execname:vals=size:sort=size' > events/uprobes/malloc/trigger
```

Example histogram on uprobe

```
# cd /sys/kernel/tracing
# echo 'hist:keys=common_pid.execname:vals=size:sort=size' > events/uprobes/malloc/trigger
# cat events/uprobes/malloc/hist
# event histogram
#
# trigger info: hist:keys=common_pid.execname:vals=hitcount,size:sort=size:size=2048 [active]
#

{ common_pid: crond           [       984] } hitcount:          1  size:         15
{ common_pid: sshd            [      1300] } hitcount:          2  size:        768
{ common_pid: bash            [      1301] } hitcount:        763  size:      19715
{ common_pid: sshd            [      1140] } hitcount:          8  size:      45824
{ common_pid: bash            [      3757] } hitcount:        418  size:     161921
{ common_pid: bash            [      3759] } hitcount:        418  size:     161921
{ common_pid: bash            [      3758] } hitcount:        418  size:     161921

Totals:
    Hits: 2028
    Entries: 7
    Dropped: 0
```

Example histogram on uprobe

```
# cd /sys/kernel/tracing
# echo 'hist:keys=common_pid.execname:vals=size:sort=size' > events/uprobes/malloc/trigger
# cat events/uprobes/malloc/hist
# event histogram
#
# trigger info: hist:keys=common_pid.execname:vals=hitcount,size:sort=size:size=2048 [active]
#

{ common_pid: crond           [       984] } hitcount:          1  size:         15
{ common_pid: sshd            [      1288] } hitcount:          2  size:        768
{ common_pid: bash            [       301] } hitcount:         53  size:      19715
{ common_pid: sshd            [      1288] } hitcount:         53  size:      45824
{ common_pid: bash            [       757] } hitcount:         18  size:     161921
{ common_pid: bash            [       759] } hitcount:         18  size:     161921
{ common_pid: bash            [      3758] } hitcount:        418  size:     161921

Totals:
    Hits: 2028
    Entries: 7
    Dropped: 0
```

BETTER

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
- Synthetic events (2018)

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
- Synthetic events (2018)
    - Connects two different events into one event

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
- Synthetic events (2018)
  - Connects two different events into one event
  - Can show latency between them

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
- Synthetic events (2018)
  - Connects two different events into one event
  - Can show latency between them
  - Fields from both events in the synthetic event

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
- Synthetic events (2018)
  - Connects two different events into one event
  - Can show latency between them
  - Fields from both events in the synthetic event
  - Uses histograms to connect the events

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
- Synthetic events (2018)
  - Connects two different events into one event
  - Can show latency between them
  - Fields from both events in the synthetic event
  - Uses histograms to connect the events
  - Can pass variables between them

What’s new? (to you!)

- Kprobe trace (2009)
- Uprobe trace (2012)
- Histograms (2016)
- Synthetic events (2018)
  - Connects two different events into one event
  - Can show latency between them
  - Fields from both events in the synthetic event
  - Uses histograms to connect the events
  - Can pass variables between them
  - Documentented in Documentation/trace/histogram.rst

Example synthetic event (wakeup latency)

```
# cd /sys/kernel/tracing
# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events
```

Example synthetic event (wakeup latency)

```
# cd /sys/kernel/tracing
# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger
```

Example synthetic event (wakeup latency)

```
# cd /sys/kernel/tracing
# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger

# echo 'hist:keys=next_pid:delta=common_timestamp.usecs-$ts1:onmatch(sched/sched_waking)'\
'trace(wakeup_lat,next_comm,next_pid,$delta)' > events/sched/sched_switch/trigger
```

Example synthetic event (wakeup latency)

```
# cd /sys/kernel/tracing
# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger

# echo 'hist:keys=next_pid:delta=common_timestamp.usecs-$ts1:onmatch(sched/sched_waking)'\
'trace(wakeup_lat,next_comm,next_pid,$delta)' > events/sched/sched_switch/trigger

# trace-cmd list -e synthetic/wakeup_lat -F --full

system: synthetic
name: wakeup_lat
ID: 1805
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:__data_loc char[] name;   offset:8;       size:8; signed:1;
        field:pid_t pid;        offset:16;       size:4; signed:1;
        field:u64 latency;      offset:24;      size:8; signed:0;

print fmt: "pid=%d, latency=%llu", REC->pid, REC->latency
```

Example synthetic event (wakeup latency)

```
# trace-cmd start -e wakeup_lat
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 2399/2399   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [000] d..4. 137919.419377: wakeup_lat: name=bash pid=1301 latency=48
          <idle>-0       [001] d..4. 137919.419610: wakeup_lat: name=kworker/u4:2 pid=3723 latency=19
    kworker/u4:2-3723    [001] d..4. 137919.419623: wakeup_lat: name=sshd pid=1300 latency=5
            sshd-1300    [001] d..4. 137919.419624: wakeup_lat: name=kworker/u4:2 pid=3723 latency=12
    kworker/u4:2-3723    [001] d..4. 137919.419684: wakeup_lat: name=sshd pid=1300 latency=2
          <idle>-0       [001] d..4. 137919.419861: wakeup_lat: name=kworker/1:1H pid=86 latency=23
          <idle>-0       [001] d..4. 137919.419913: wakeup_lat: name=rcu_preempt pid=14 latency=3
          <idle>-0       [001] d..4. 137919.423958: wakeup_lat: name=rcu_preempt pid=14 latency=4
          <idle>-0       [001] d..4. 137919.424961: wakeup_lat: name=kworker/1:1 pid=1223 latency=5
          <idle>-0       [001] d..4. 137919.427909: wakeup_lat: name=rcu_preempt pid=14 latency=4
          <idle>-0       [001] d..4. 137919.431913: wakeup_lat: name=rcu_preempt pid=14 latency=5
          <idle>-0       [001] d..4. 137919.432976: wakeup_lat: name=rcu_preempt pid=14 latency=18
          <idle>-0       [001] d..4. 137919.437016: wakeup_lat: name=rcu_preempt pid=14 latency=9
          <idle>-0       [000] d..4. 137919.445260: wakeup_lat: name=kworker/0:1 pid=3727 latency=59
          <idle>-0       [001] d..4. 137919.605089: wakeup_lat: name=kcompactd0 pid=28 latency=23
          <idle>-0       [000] d..4. 137919.653021: wakeup_lat: name=kworker/0:1 pid=3727 latency=18
```

Example synthetic event with histogram

```
# cd /sys/kernel/tracing
# echo 'hist:keys=name,latency.buckets=10:sort=name,latency' > events/synthetic/wakeup_lat/trigger
```

Example synthetic event with histogram

```
# cd /sys/kernel/tracing
# echo 'hist:keys=name,latency.buckets=10:sort=name,latency' > events/synthetic/wakeup_lat/trigger
# cat events/synthetic/wakeup/hist
# event histogram
#
# trigger info: hist:keys=name,latency.buckets=10:vals=hitcount:sort=name,latency.buckets=10:size=2048 [active]
#

{ name: bash                                          , latency: ~ 80-89 } hitcount:          2
{ name: bash                                          , latency: ~ 190-199 } hitcount:          1
{ name: bash                                          , latency: ~ 200-209 } hitcount:          1
{ name: kworker/0:1                                   , latency: ~ 10-19 } hitcount:          2
{ name: kworker/u4:0                                  , latency: ~ 70-79 } hitcount:          1
{ name: kworker/u4:0                                  , latency: ~ 80-89 } hitcount:          1
{ name: kworker/u4:0                                  , latency: ~ 170-179 } hitcount:          1
{ name: kworker/u4:0                                  , latency: ~ 74860-74869 } hitcount:          1
{ name: migration/0                                   , latency: ~ 60-69 } hitcount:          1
{ name: migration/1                                   , latency: ~ 70-79 } hitcount:          1
{ name: rcu_preempt                                   , latency: ~ 0-9 } hitcount:          1
{ name: rcu_preempt                                   , latency: ~ 10-19 } hitcount:          3
{ name: rcu_preempt                                   , latency: ~ 20-29 } hitcount:         16
{ name: rcu_preempt                                   , latency: ~ 30-39 } hitcount:         13
{ name: rcu_preempt                                   , latency: ~ 150-159 } hitcount:          1
{ name: sshd                                          , latency: ~ 10-19 } hitcount:          2
{ name: sshd                                          , latency: ~ 20-29 } hitcount:          2
{ name: sshd                                          , latency: ~ 70-79 } hitcount:          2
{ name: sshd                                          , latency: ~ 80-89 } hitcount:          1

Totals:
    Hits: 86
    Entries: 38
    Dropped: 0
```

Example synthetic event (wakeup latency)

```
# cd /sys/kernel/tracing
# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger

# echo 'hist:keys=next_pid:delta=common_timestamp.usecs-$ts1:onmatch(sched/sched_waking)'\
'trace(wakeup_lat,next_comm,next_pid,$delta)' > events/sched/sched_switch/trigger
```

Example synthetic event (wakeup latency)

```
# cd /sys/kernel/tracing
# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger

# echo 'hist:keys=next_pid:delta=common_timestamp.usecs-$ts1:onmatch(sched/sched_waking)'\
'trace(wakeup_lat,next_comm,next_pid,$delta)' > events/sched/sched_switch/trigger
```

TRIVIAL!

Example synthetic event (wakeup latency)

```
# cd /sys/kernel/tracing
# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger

# echo 'hist:keys=next_pid:delta=common_timestamp.usecs-$ts1:onmatch(sched/sched_waking)'\
'trace(wakeup_lat,next_comm,next_pid,$delta)' > events/sched/sched_switch/trigger
```

YEAH

RIGHT!

What’s new?

What’s new? (Since the pandemic started)

What’s new? (Since the pandemic started)

- libtracefs (2020)

What’s new? (Since the pandemic started)

- libtracefs (2020)
  - Interface for everything needed in the tracefs file system

What’s new? (Since the pandemic started)

- libtracefs (2020)
  - Interface for everything needed in the tracefs file system
    - (Well almost everything)

What’s new? (Since the pandemic started)

- libtracefs (2020)
  - Interface for everything needed in the tracefs file system
    - (Well almost everything)
  - Written in C

What’s new? (Since the pandemic started)

- libtracefs (2020)
  - Interface for everything needed in the tracefs file system
    - (Well almost everything)
  - Written in C
  - Interfaces to make histograms, kprobes, uprobes and synthetic events

What’s new? (Since the pandemic started)

- libtracefs (2020)
  - Interface for everything needed in the tracefs file system
    - (Well almost everything)
  - Written in C
  - Interfaces to make histograms, kprobes, uprobes and synthetic events
  - Full man pages (https://www.trace-cmd.org/Documentation/libtracefs/libtracefs.html)

What’s new? (Since the pandemic started)

- libtracefs (2020)
  - Interface for everything needed in the tracefs file system
    - (Well almost everything)
  - Written in C
  - Interfaces to make histograms, kprobes, uprobes and synthetic events
  - Full man pages (https://www.trace-cmd.org/Documentation/libtracefs/libtracefs.html)
    - With examples

What’s new? (Since the pandemic started)

- libtracefs (2020)
  - Interface for everything needed in the tracefs file system
    - (Well almost everything)
  - Written in C
  - Interfaces to make histograms, kprobes, uprobes and synthetic events
  - Full man pages (https://www.trace-cmd.org/Documentation/libtracefs/libtracefs.html)
    - With examples
    - tracefs_sq() has an example that creates sqlhist application

What’s new? (Since the pandemic started)

- libtracefs (2020)
  - Interface for everything needed in the tracefs file system
    - (Well almost everything)
  - Written in C
  - Interfaces to make histograms, kprobes, uprobes and synthetic events
  - Full man pages (https://www.trace-cmd.org/Documentation/libtracefs/libtracefs.html)
    - With examples
    - tracefs_sq() has an example that creates sqlhist application
    - sqlhist has it’s own man page too

Example synthetic event (wakeup latency)

```
# cd /sys/kernel/tracing
# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger

# echo 'hist:keys=next_pid:delta=common_timestamp.usecs-$ts1:onmatch(sched/sched_waking)'\
'trace(wakeup_lat,next_comm,next_pid,$delta)' > events/sched/sched_switch/trigger
```

Example synthetic event with sqlhist

```
# sqlhist -n wakeup_lat 'SELECT end.next_comm AS name, start.pid,
      (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS latency
      FROM sched_waking AS start JOIN sched_switch AS end ON start.pid = end.next_pid'
```

Example synthetic event with sqlhist

```
# sqlhist -n wakeup_lat 'SELECT end.next_comm AS name, start.pid,
      (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS latency
      FROM sched_waking AS start JOIN sched_switch AS end ON start.pid = end.next_pid'
```

Example synthetic event with sqlhist

```
# sqlhist -n wakeup_lat 'SELECT end.next_comm AS name, start.pid,
      (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS latency
      FROM sched_waking AS start JOIN sched_switch AS end ON start.pid = end.next_pid'
```

Example synthetic event with sqlhist

```
# sqlhist -n wakeup_lat 'SELECT end.next_comm AS name, start.pid,
      (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS latency
      FROM sched_waking AS start JOIN sched_switch AS end ON start.pid = end.next_pid'
```

Example synthetic event with sqlhist

```
# sqlhist -n wakeup_lat 'SELECT end.next_comm AS name, start.pid,
      (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS latency
      FROM sched_waking AS start JOIN sched_switch AS end ON start.pid = end.next_pid'

# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events
```

Example synthetic event with sqlhist

```
# sqlhist -n wakeup_lat 'SELECT end.next_comm AS name, start.pid,
      (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS latency
      FROM sched_waking AS start JOIN sched_switch AS end ON start.pid = end.next_pid'

# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger
```

Example synthetic event with sqlhist

```
# sqlhist -n wakeup_lat 'SELECT end.next_comm AS name, start.pid,
      (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS latency
      FROM sched_waking AS start JOIN sched_switch AS end ON start.pid = end.next_pid'

# echo 'wakeup_lat char name[]; pid_t pid; u64 latency' > synthetic_events

# echo 'hist:keys=pid:ts1=common_timestamp.usecs' > events/sched/sched_waking/trigger

# echo 'hist:keys=next_pid:delta=common_timestamp.usecs-$ts1:onmatch(sched/sched_waking)'\
'trace(wakeup_lat,next_comm,next_pid,$delta)' > events/sched/sched_switch/trigger
```

Fun with sqlhist (what system calls block the longest?)

Fun with sqlhist (what system calls block the longest?)

A Harald Seiler request

Fun with sqlhist (what system calls block the longest?)

```
# sqlhist -e -n sysname SELECT start.id, end.prev_pid FROM sys_enter AS start
      JOIN sched_switch AS end ON start.common_pid = end.prev_pid
      WHERE end.prev_state == 2'
```

Fun with sqlhist (what system calls block the longest?)

```
# sqlhist -e -n sysname SELECT start.id, end.prev_pid FROM sys_enter AS start
      JOIN sched_switch AS end ON start.common_pid = end.prev_pid
      WHERE end.prev_state == 2'

# sqlhist -e -n offcpu 'SELECT start.id, end.next_comm AS comm, end.next_pid AS pid,
    (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS lat FROM sysname AS start
    JOIN sched_switch AS end ON start.prev_pid = end.next_pid'
```

Fun with sqlhist (what system calls block the longest?)

```
# sqlhist -e -n sysname SELECT start.id, end.prev_pid FROM sys_enter AS start
      JOIN sched_switch AS end ON start.common_pid = end.prev_pid
      WHERE end.prev_state == 2'

# sqlhist -e -n offcpu 'SELECT start.id, end.next_comm AS comm, end.next_pid AS pid,
    (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS lat FROM sysname AS start
    JOIN sched_switch AS end ON start.prev_pid = end.next_pid'

# cd /sys/kernel/tracing
# echo 'hist:keys=id.syscall,comm,pid:vals=lat' > events/synthetic/offcpu/trigger
```

Fun with sqlhist (what system calls block the longest?)

```
# sqlhist -e -n sysname SELECT start.id, end.prev_pid FROM sys_enter AS start
      JOIN sched_switch AS end ON start.common_pid = end.prev_pid
      WHERE end.prev_state == 2'

# sqlhist -e -n offcpu 'SELECT start.id, end.next_comm AS comm, end.next_pid AS pid,
    (end.TIMESTAMP_USECS - start.TIMESTAMP_USECS) AS lat FROM sysname AS start
    JOIN sched_switch AS end ON start.prev_pid = end.next_pid'

# cd /sys/kernel/tracing
# echo 'hist:keys=id.syscall,comm,pid:vals=lat' > events/synthetic/offcpu/trigger
# cat events/synthetic/offcpu/hist
# event histogram
#
# trigger info: hist:keys=id.syscall,comm,pid:vals=hitcount,lat:sort=hitcount:size=2048 [active]
#

{ id: sys_munmap         [ 11], comm: nm-dispatcher     , pid:   4172 } hitcount:    1  lat:       64
{ id: sys_rt_sigaction   [ 13], comm: nm-dispatcher     , pid:   4171 } hitcount:    1  lat:       67
{ id: sys_select         [ 23], comm: sshd              , pid:   1153 } hitcount:    1  lat:       19
{ id: sys_munmap         [ 11], comm: nm-dhcp-helper    , pid:   4167 } hitcount:    1  lat:       31
{ id: sys_fsync          [ 74], comm: dhclient          , pid:   3777 } hitcount:    2  lat:    85876
{ id: sys_fsync          [ 74], comm: systemd-journal   , pid:    570 } hitcount:    2  lat:    79773
{ id: sys_futex          [202], comm: nm-dhcp-helper    , pid:   4166 } hitcount:    3  lat:       21

Totals:
    Hits: 11
    Entries: 7
    Dropped: 0
```

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
  - Limit what an event shows (save space on ring buffer)

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
  - Limit what an event shows (save space on ring buffer)
  - Extend trace events like kprobes

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
  - Limit what an event shows (save space on ring buffer)
  - Extend trace events like kprobes
  - Needs documentation!

Find network event with skbuff

```
# trace-cmd list -e netif_receive_skb -F

system: net
name: netif_receive_skb
ID: 1499
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 * skbaddr;   offset:8;       size:8; signed:0;
        field:unsigned int len; offset:16;      size:4; signed:0;
        field:__data_loc char[] name;   offset:20;      size:4; signed:1;
```

Remember kprobe trace example?

```
# echo 'p:ip_rcv ip_rcv_core skb=$arg1 dev=+0(+0x10($arg1)):string' > /sys/kernel/tracing/kprobe_events
```

Example eprobe trace on network event

```
# trace-cmd reset
# cd /sys/kernel/tracing
# echo 'e:netdev net/netif_receive_skb dev=+0(+0x10($skbaddr)):string' > dynamic_events
# trace-cmd list -e eprobes -F --full

system: eprobes
name: netdev
ID: 1810
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:__data_loc char[] dev;    offset:8;       size:4; signed:1;
```

Example eprobe trace

```
# trace-cmd start -e netdev
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 20/20   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          <idle>-0       [001] .Ns2. 142503.448784: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142503.448809: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142503.823755: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142503.825251: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142503.913169: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142503.913309: netdev: (net.netif_receive_skb) dev="enp1s0"
            sshd-1140    [001] ..s2. 142503.913404: netdev: (net.netif_receive_skb) dev="lo"
            sshd-1300    [000] ..s2. 142503.913406: netdev: (net.netif_receive_skb) dev="lo"
            sshd-1140    [001] ..s2. 142503.913421: netdev: (net.netif_receive_skb) dev="lo"
            sshd-1300    [000] ..s2. 142503.913425: netdev: (net.netif_receive_skb) dev="lo"
          <idle>-0       [001] ..s2. 142503.979837: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142503.981620: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142504.020107: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142504.021010: netdev: (net.netif_receive_skb) dev="enp1s0"
          <idle>-0       [001] ..s2. 142504.101074: netdev: (net.netif_receive_skb) dev="enp1s0"
```

Example eprobe for seeing openat system call files

```
# trace-cmd list -e sys_enter_openat -F

system: syscalls
name: sys_enter_openat2
ID: 646
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:int __syscall_nr; offset:8;       size:4; signed:1;
        field:int dfd;  offset:16;      size:8; signed:0;
        field:const char * filename;    offset:24;      size:8; signed:0;
        field:struct open_how * how;    offset:32;      size:8; signed:0;
        field:size_t usize;     offset:40;      size:8; signed:0;
```

Example eprobe for seeing openat system call files

```
# trace-cmd reset
# cd /sys/kernel/tracing
# echo 'e:open syscalls/sys_enter_openat file=+0($filename):ustring' > dynamic_events
# trace-cmd list -e eprobes -F --full
```

Example eprobe trace

```
# trace-cmd start -e open
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 136/136   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
       trace-cmd-12600   [005] ...1. 151293.992847: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/tls/x86_64/x86_64/librt
       trace-cmd-12600   [005] ...1. 151293.992856: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/tls/x86_64/librt.so.1"
            less-12601   [002] ...1. 151293.992859: open: (syscalls.sys_enter_openat) file="/etc/ld.so.cache"
       trace-cmd-12600   [005] ...1. 151293.992862: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/tls/x86_64/librt.so.1"
       trace-cmd-12600   [005] ...1. 151293.992867: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/tls/librt.so.1"
       trace-cmd-12600   [005] ...1. 151293.992872: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/x86_64/x86_64/librt.so.
       trace-cmd-12600   [005] ...1. 151293.992878: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/x86_64/librt.so.1"
       trace-cmd-12600   [005] ...1. 151293.992883: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/x86_64/librt.so.1"
            less-12601   [002] ...1. 151293.992887: open: (syscalls.sys_enter_openat) file="/lib64/libtinfo.so.6"
       trace-cmd-12600   [005] ...1. 151293.992889: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/librt.so.1"
       trace-cmd-12600   [005] ...1. 151293.992896: open: (syscalls.sys_enter_openat) file="/etc/ld.so.cache"
       trace-cmd-12600   [005] ...1. 151293.992924: open: (syscalls.sys_enter_openat) file="/lib64/librt.so.1"
            less-12601   [002] ...1. 151293.992959: open: (syscalls.sys_enter_openat) file="/lib64/libc.so.6"
       trace-cmd-12600   [005] ...1. 151293.992988: open: (syscalls.sys_enter_openat) file="/usr/local/lib64/libpthread.so.0"
       trace-cmd-12600   [005] ...1. 151293.992995: open: (syscalls.sys_enter_openat) file="/lib64/libpthread.so.0"
```

Example eprobe trace

```
ls-12606   [002] ...1. 151304.076524: open: (syscalls.sys_enter_openat) file="/etc/ld.so.cache"
ls-12606   [002] ...1. 151304.076558: open: (syscalls.sys_enter_openat) file="/lib64/libselinux.so.1"
ls-12606   [002] ...1. 151304.076633: open: (syscalls.sys_enter_openat) file="/lib64/libcap.so.2"
ls-12606   [002] ...1. 151304.076686: open: (syscalls.sys_enter_openat) file="/lib64/libc.so.6"
ls-12606   [002] ...1. 151304.076766: open: (syscalls.sys_enter_openat) file="/lib64/libpcre2-8.so.0"
ls-12606   [002] ...1. 151304.076824: open: (syscalls.sys_enter_openat) file="/lib64/libdl.so.2"
ls-12606   [002] ...1. 151304.076878: open: (syscalls.sys_enter_openat) file="/lib64/libpthread.so.0"
ls-12606   [002] ...1. 151304.077397: open: (syscalls.sys_enter_openat) file="/proc/filesystems"
ls-12606   [002] ...1. 151304.077476: open: (syscalls.sys_enter_openat) file="/usr/lib/locale/locale-archive"
ls-12606   [002] ...1. 151304.077568: open: (syscalls.sys_enter_openat) file="/usr/share/locale/locale.alias"
ls-12606   [002] ...1. 151304.077625: open: (syscalls.sys_enter_openat) file="/usr/share/locale/en_US.UTF-8/LC_TIME/co
ls-12606   [002] ...1. 151304.077631: open: (syscalls.sys_enter_openat) file="/usr/share/locale/en_US.utf8/LC_TIME/cor
ls-12606   [002] ...1. 151304.077635: open: (syscalls.sys_enter_openat) file="/usr/share/locale/en_US/LC_TIME/coreutil
ls-12606   [002] ...1. 151304.077650: open: (syscalls.sys_enter_openat) file="/usr/share/locale/en.UTF-8/LC_TIME/coreu
ls-12606   [002] ...1. 151304.077655: open: (syscalls.sys_enter_openat) file="/usr/share/locale/en.utf8/LC_TIME/coreut
ls-12606   [002] ...1. 151304.077659: open: (syscalls.sys_enter_openat) file="/usr/share/locale/en/LC_TIME/coreutils.
ls-12606   [002] ...1. 151304.077673: open: (syscalls.sys_enter_openat) file="/usr/lib64/gconv/gconv-modules.cache"
ls-12606   [002] ...1. 151304.077733: open: (syscalls.sys_enter_openat) file="."
ls-12606   [002] ...1. 151304.077880: open: (syscalls.sys_enter_openat) file="/etc/nsswitch.conf"
ls-12606   [002] ...1. 151304.077918: open: (syscalls.sys_enter_openat) file="/etc/ld.so.cache"
ls-12606   [002] ...1. 151304.077950: open: (syscalls.sys_enter_openat) file="/lib64/libnss_files.so.2"
ls-12606   [002] ...1. 151304.078060: open: (syscalls.sys_enter_openat) file=(fault)
ls-12606   [002] ...1. 151304.078162: open: (syscalls.sys_enter_openat) file="/etc/group"
ls-12606   [002] ...1. 151304.078403: open: (syscalls.sys_enter_openat) file="/usr/share/locale/en_US.UTF-8/LC_MESSAGE
ls-12606   [002] ...1. 151304.078409: open: (syscalls.sys_enter_openat)
```

Simple “open” program

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>

static const char *file = "/etc/passwd";

int main (int argc, char **argv)
{
        int fd;

        fd = open(file, O_RDONLY);
        if (fd < 0)
                perror(file);
        close(fd);
        return 0;
}
```

Example eprobe trace

```
# trace-cmd start -e open -F openat
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 136/136   #P:2
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          openat-12625   [006] ...1. 151721.857580: open: (syscalls.sys_enter_openat) file="/etc/ld.so.cache"
          openat-12625   [006] ...1. 151721.857612: open: (syscalls.sys_enter_openat) file="/lib64/libc.so.6"
          openat-12625   [006] ...1. 151721.857879: open: (syscalls.sys_enter_openat) file=(fault)
```

Simple “open” program

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>

static const char *file = "/etc/passwd";

int main (int argc, char **argv)
{
        int fd;

        fd = open(file, O_RDONLY);
        if (fd < 0)
                perror(file);
        close(fd);
        return 0;
}
```

Simple “open” program

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>

static const char *file = "/etc/passwd";

int main (int argc, char **argv)
{
        int fd;

        fd = open(file, O_RDONLY);
        if (fd < 0)
                perror(file);
        close(fd);
        return 0;
}
```

Trace event at the return of the openat system call

```
# trace-cmd list -e sys_enter_openat -F

system: syscalls
name: sys_exit_openat
ID: 647
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:int __syscall_nr; offset:8;       size:4; signed:1;
        field:long ret; offset:16;      size:8; signed:1;
```

Example sqlhist for seeing open files

```
# sqlhist -e -n myopen 'SELECT start.filename, end.ret
      FROM sys_enter_openat AS start JOIN sys_exit_openat AS end
      ON start.common_pid = end.common_pid'

# trace-cmd list -e myopen -F

system: synthetic
name: myopen
ID: 1710
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:u64 filename;     offset:8;   size:8;     signed:0;
      field:s64 ret;    offset:16;  size:8;     signed:1;
```

Example eprobe for seeing openat system call files

```
# cd /sys/kernel/tracing
# echo 'e:open synthetic/myopen file=+0($filename):ustring ret=$ret' > dynamic_events
# trace-cmd list -e eprobes -F --full
system: eprobes
name: open
ID: 1711
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:__data_loc char[] file; offset:8;   size:4;     signed:1;
      field:u64 ret;    offset:12;  size:8;     signed:0;
```

Example eprobe trace

```
# trace-cmd start -e open -F openat
# trace-cmd show
# tracer: nop
#
# entries-in-buffer/entries-written: 3/3   #P:8
#
#                                _-----=> irqs-off/BH-disabled
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / _-=> migrate-disable
#                              |||| /     delay
#           TASK-PID     CPU#  |||||  TIMESTAMP  FUNCTION
#              | |         |   |||||     |         |
          openat-13174   [002] ...3. 157975.394662: open: (synthetic.myopen) file="/etc/ld.so.cache" ret=0x3
          openat-13174   [002] ...3. 157975.394662: open: (synthetic.myopen) file="/lib64/libc.so.6" ret=0x3
          openat-13174   [002] ...3. 157975.394662: open: (synthetic.myopen) file="/etc/passwd" ret=0x3
```

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
  - Extend the kernel command line

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
  - Extend the kernel command line
  - Attaches to the init ramdisk

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
  - Extend the kernel command line
  - Attaches to the init ramdisk
  - Attaches to vmlinux (2022, 5.19)

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
  - Extend the kernel command line
  - Attaches to the init ramdisk
  - Attaches to vmlinux (2022, 5.19)
  - Json format

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
  - Extend the kernel command line
  - Attaches to the init ramdisk
  - Attaches to vmlinux (2022, 5.19)
  - Json format
  - Located in tools/bootconfig

bootconfig example file

```
ftrace {
        tracer = function_graph;
        options = event-fork, sym-addr, stacktrace;
        buffer_size = 1M;
        alloc_snapshot;
        trace_clock = global;
        events = "task:task_newtask", "initcall:*";
        event.sched.sched_process_exec {
                filter = "pid < 128";
        }
        instance.bar {
                event.kprobes {
                        myevent {
                                probes = "vfs_read $arg2 $arg3";
                        }
                        myevent2 {
                                probes = "vfs_write $arg2 +0($arg2):ustring $arg3";
                        }
                        myevent3 {
                                probes = "initrd_load";
                        }
                        enable
                }
        }
        instance.foo {
                tracer = function;
                tracing_on = false;
        };
}
kernel {
        ftrace_dump_on_oops = "orig_cpu"
        traceoff_on_warning
}
```

bootconfig example file (init ramdisk)

```
# bootconfig -a /work/git/bootconfigs/tracing.bconf -e /boot/initrd.img
```

bootconfig example file (embedded)

make menuconfig
→ General setup

```
-*- Boot config support
[*]   Embed bootconfig file in the kernel
(/work/git/bootconfigs/tracing.bconf) Embedded bootconfig file path
```

→ Processor type and features

```
[*] Built-in kernel command line
(bootconfig) Built-in kernel command string
```

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
- CUSTOM_TRACE_EVENT macro (2022)

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
- CUSTOM_TRACE_EVENT macro (2022)
    - Where modules can redefine an existing trace event

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
- CUSTOM_TRACE_EVENT macro (2022)
  - Where modules can redefine an existing trace event
  - See samples/trace_events/trace_custom_sched.h

TRACE_EVENT(sched_switch)

```
TRACE_EVENT(sched_switch,

        TP_PROTO(bool preempt,
                 unsigned int prev_state,
                 struct task_struct *prev,
                 struct task_struct *next),

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

sched_switch trace event

```
# trace-cmd list -e sched_switch -F

system: sched
name: sched_switch
ID: 308
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[TASK_COMM_LEN];    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[TASK_COMM_LEN];    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;
```

CUSTOM_TRACE_EVENT(sched_switch)

```
TRACE_CUSTOM_EVENT(sched_switch,

        TP_PROTO(bool preempt,
                 unsigned int prev_state,
                 struct task_struct *prev,
                 struct task_struct *next),

        TP_ARGS(preempt, prev_state, prev, next),

        TP_STRUCT__entry(
                __field(        unsigned short,         prev_prio       )
                __field(        unsigned short,         next_prio       )
                __field(        pid_t,  next_pid                        )
        ),

        TP_fast_assign(
                __entry->prev_prio      = prev->prio;
                __entry->next_pid       = next->pid;
                __entry->next_prio      = next->prio;
        ),

        TP_printk("prev_prio=%d next_pid=%d next_prio=%d",
                  __entry->prev_prio, __entry->next_pid, __entry->next_prio)
)
```

custom:sched_switch trace event

```
# trace-cmd list -e custom:sched_switch -F

system: custom
name: sched_switch
ID: 1708
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:unsigned short prev_prio; offset:8;       size:2; signed:0;
        field:unsigned short next_prio; offset:10;      size:2; signed:0;
        field:pid_t next_pid;   offset:12;      size:4; signed:1;
```

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
- CUSTOM_TRACE_EVENT macro (2022)
- New latency tracers (2022)

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
- CUSTOM_TRACE_EVENT macro (2022)
- New latency tracers (2022)
  - timerlat

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
- CUSTOM_TRACE_EVENT macro (2022)
- New latency tracers (2022)
  - timerlat
  - osnoise

What’s new? (Since the pandemic started)

- libtracefs (2020)
- Event probes (eprobes) (2021)
- bootconfig (2020)
- CUSTOM_TRACE_EVENT macro (2022)
- New latency tracers (2022)
  - timerlat
  - osnoise
  - Superseeds hwlat tracer

Thank you!