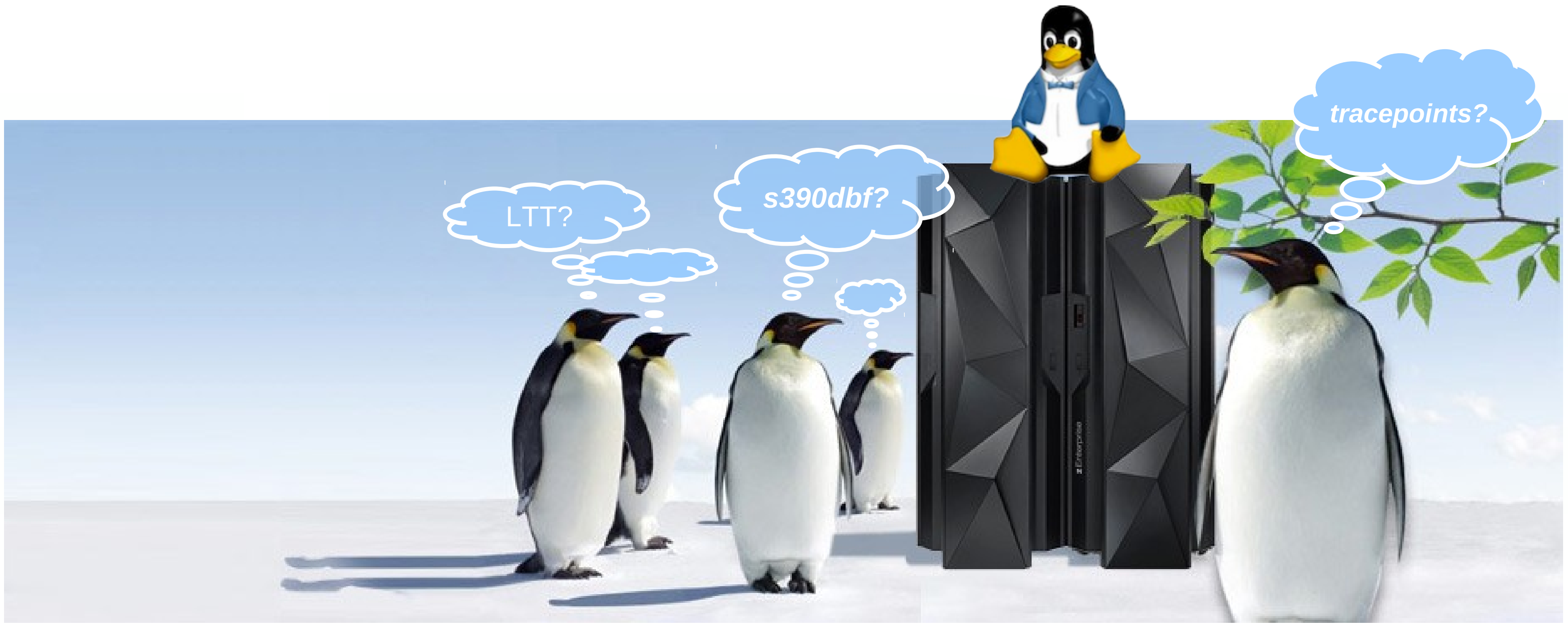
Michael Holzheu

LINUXCON Düsseldorf 15/10/2014

Kernel Event Tracing

on the Mainframe



Trademarks & Disclaimer

Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs, and IFLs)

Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at

www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”).

No other workload processing is authorized for execution on an SE.

IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Introduction

S390 debug feature

Tracepoints

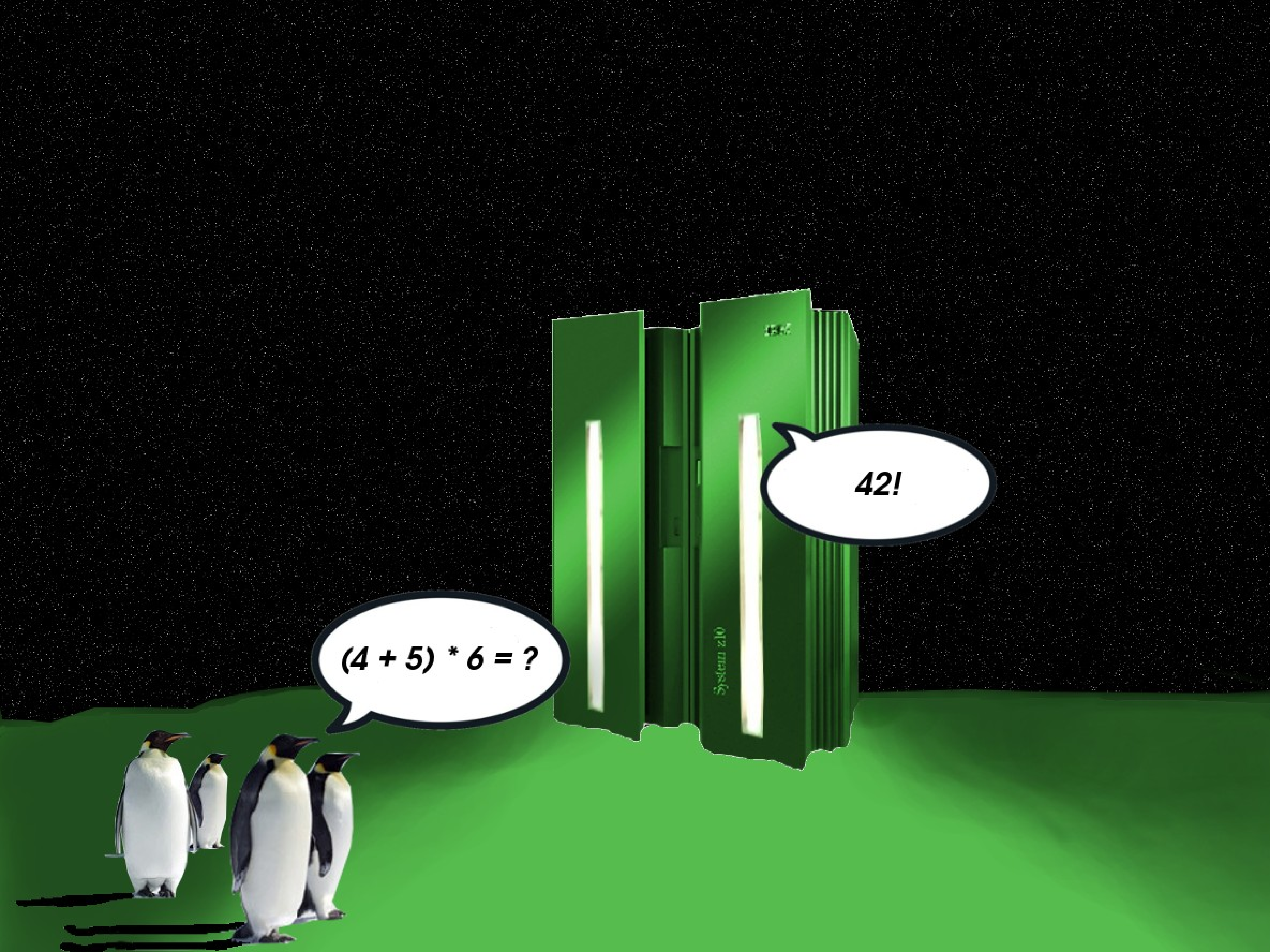
Comparison & Outlook

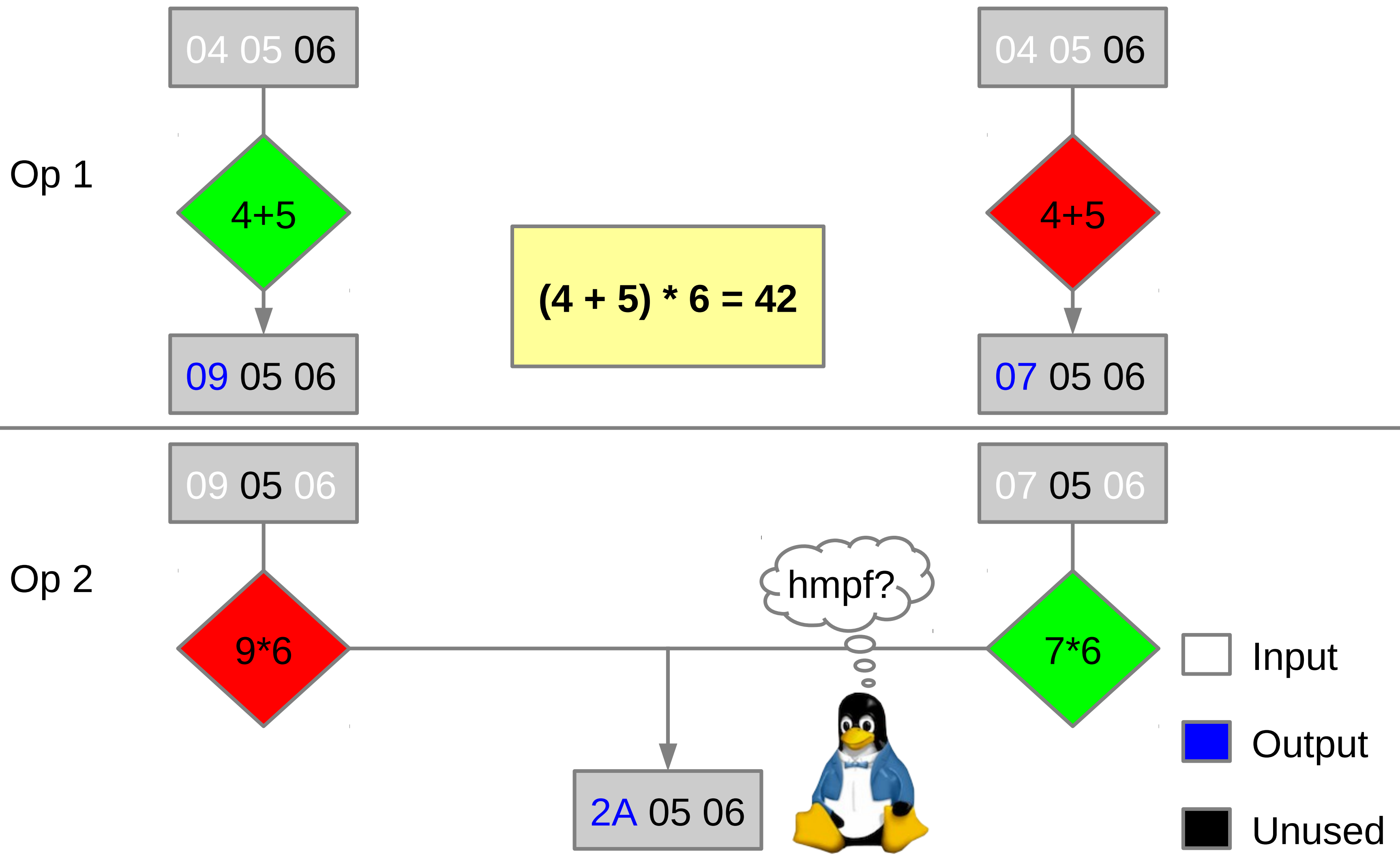
Introduction - What is tracing?

http://en.wikipedia.org/wiki/Tracing_(software)

In software engineering, tracing is a ***specialized use of logging*** to ***record information*** about a ***program's execution***. This information is typically ***used by programmers*** for ***debugging purposes***, and additionally, ... by ... technical support personnel and software ***monitoring tools*** to ***diagnose*** common ***problems*** with software.

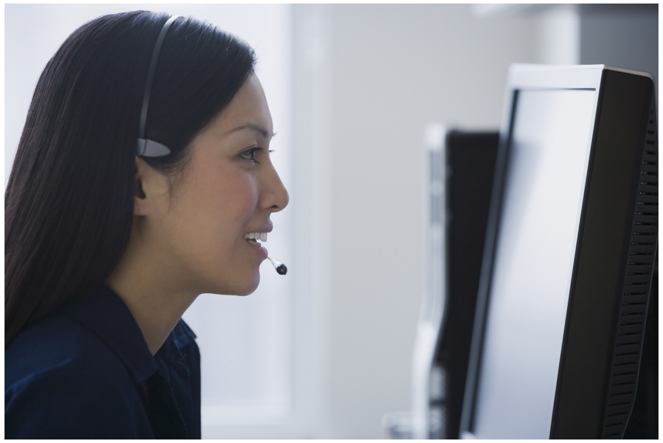
Scenario


42!
(4 + 5) * 6 = ?
IBM
System z10


Op 1
04 05 06
4+5
09 05 06
04 05 06
4+5
07 05 06
(4 + 5) * 6 = 42
Op 2
09 05 06
9*6
07 05 06
7*6
hmpf?
2A 05 06
Input
Output
Unused

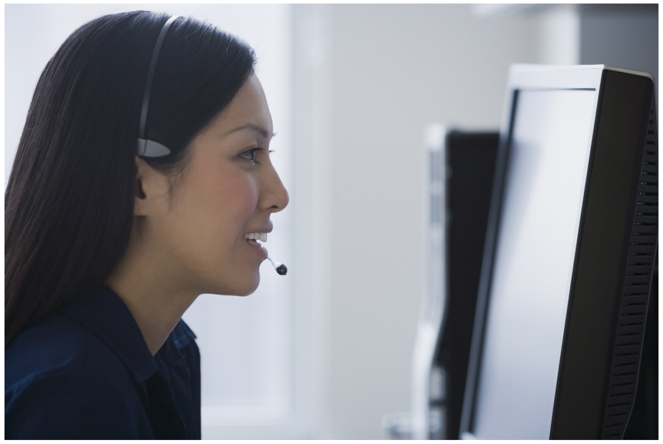
IBM Service Process...

Hmm,
42 looks wrong to me!

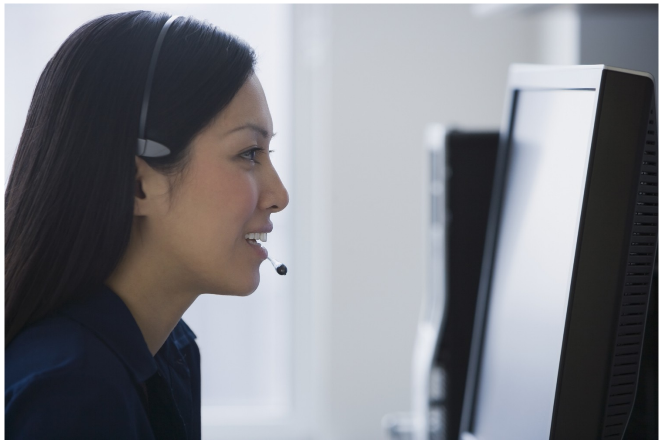


Hello IBM?

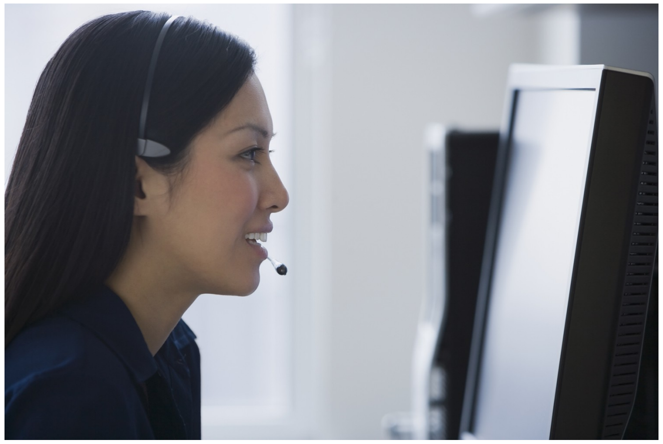
Let me check:
Yes, (4+5)*6=42
is wrong!

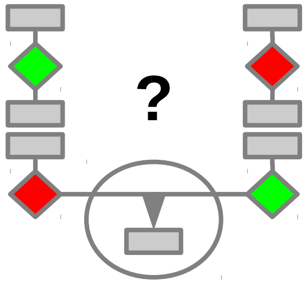


I will contact our experts!

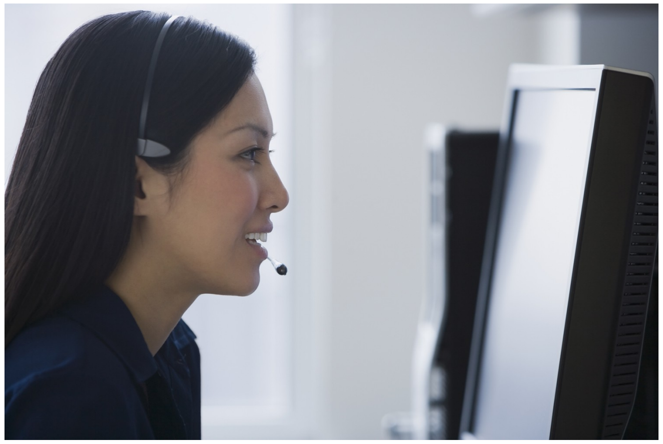


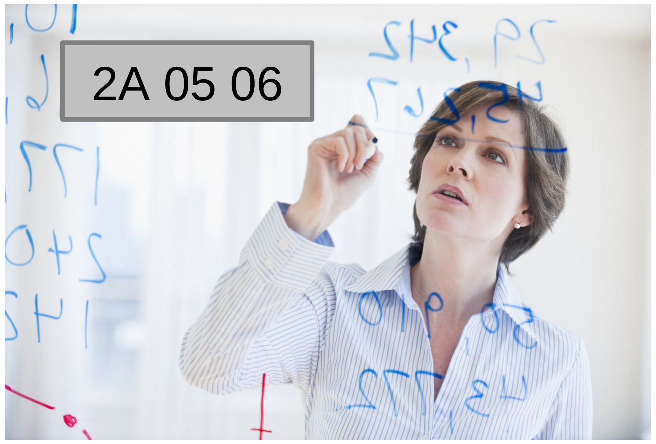
(IBM Lab)




?

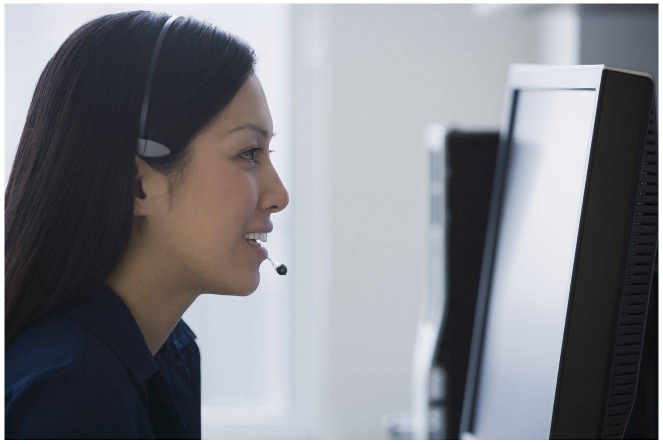
I need more information!
(IBM Lab)



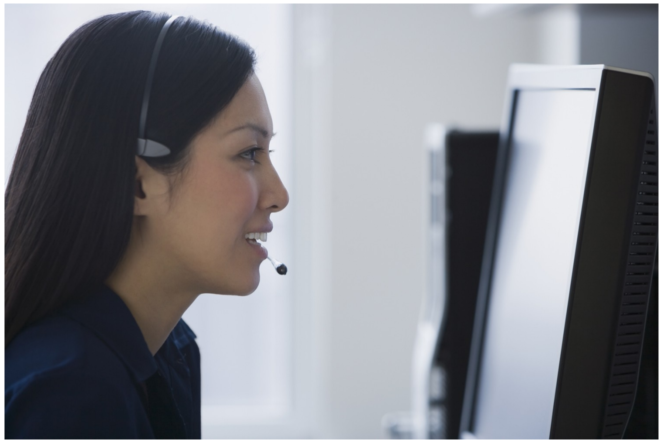

2A 05 06

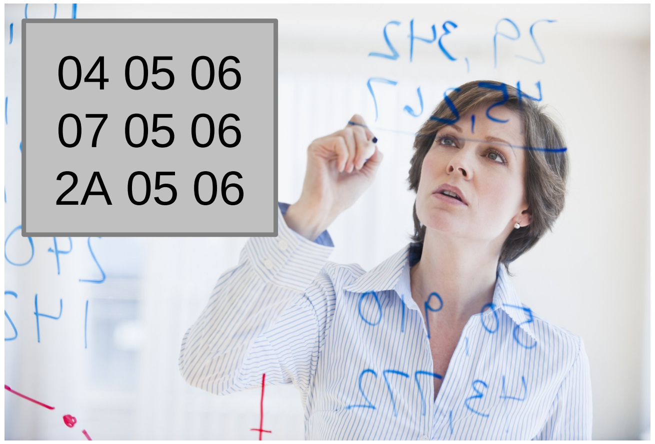
Could you send us a trace?

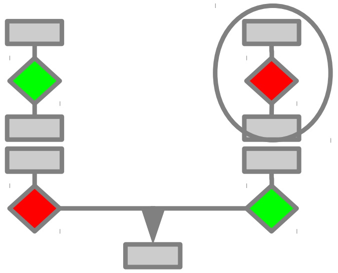
(IBM Lab)

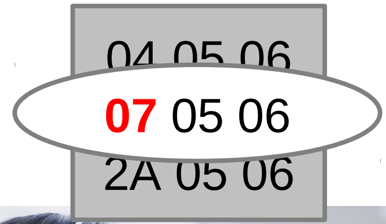


(IBM Lab)



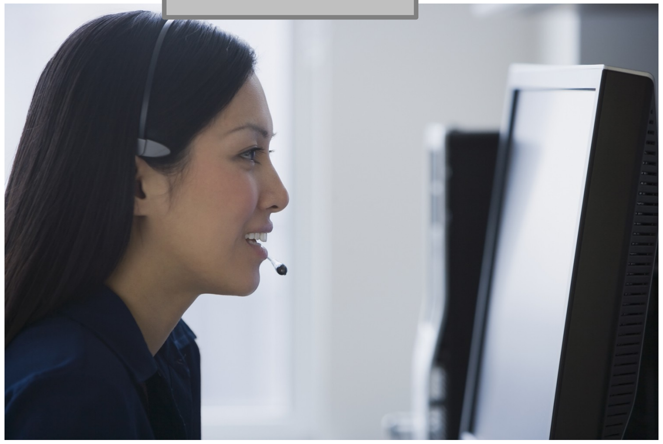






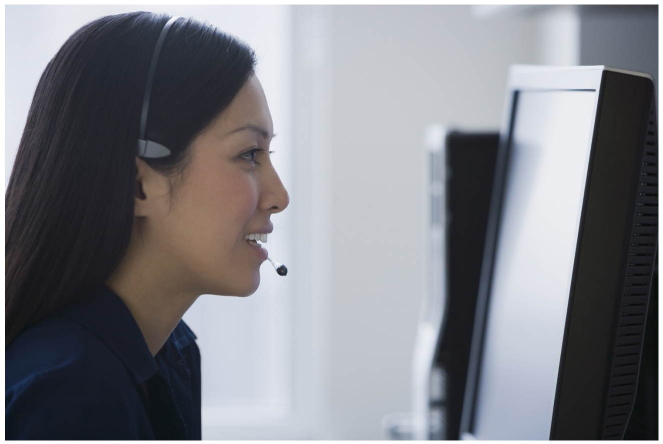
(IBM Lab)

Problem in addition



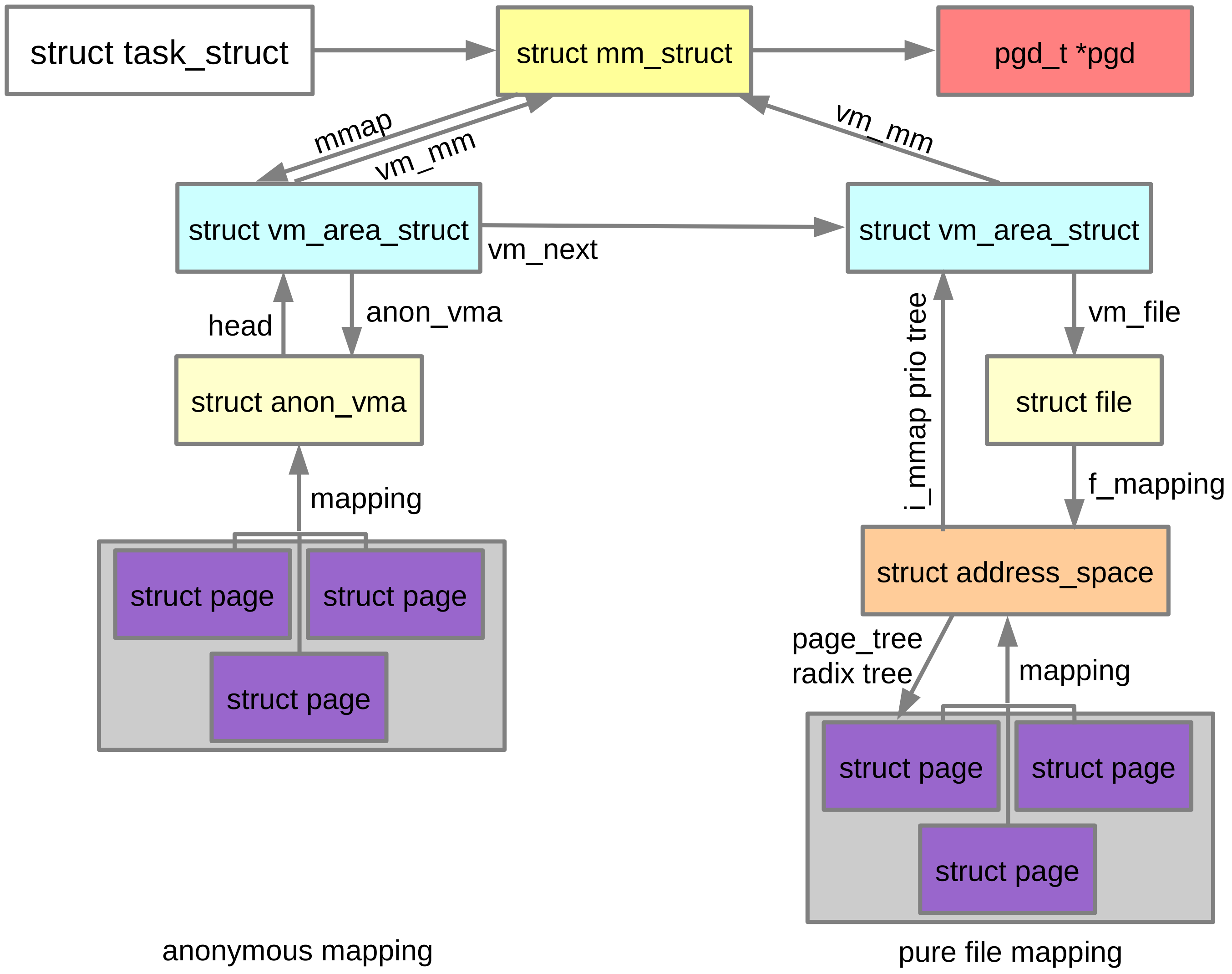
Fix

Great job! Thank you!



Too simple?

Linux memory management



Introduction - Tracing Basics

- **Purpose:**
  - Find bugs and malfunctions
  - Performance analysis
- **Root cause of problem is often far away from problem manifestation**
  - Example: FCP adapter corrupts SCSI payload
- **Mechanisms:**
  - Static vs. dynamic
- **How is it used:**
  - Live debugging
  - First Failure Data Capture (FFDC)
- **Main use case for s390dbf tracing on mainframe:**
  - Static tracing for FFDC

Introduction - Challenges of tracing

- **Do not waste resources:**
  - CPU
  - Memory
  - Disk
- **Achieve high Information density**
- **Make trace data easy consumable**
  - For humans
  - For machines
- **Make trace data persistent**
- **Support multiple components**
- **Isolate components**

Static kernel tracing

2000: Linux Trace Toolkit (LTT)

2000: s390 debug feature (s390dbf)

2005: Linux Trace Toolkit Next Generation (LTTng)

2008: Tracepoints

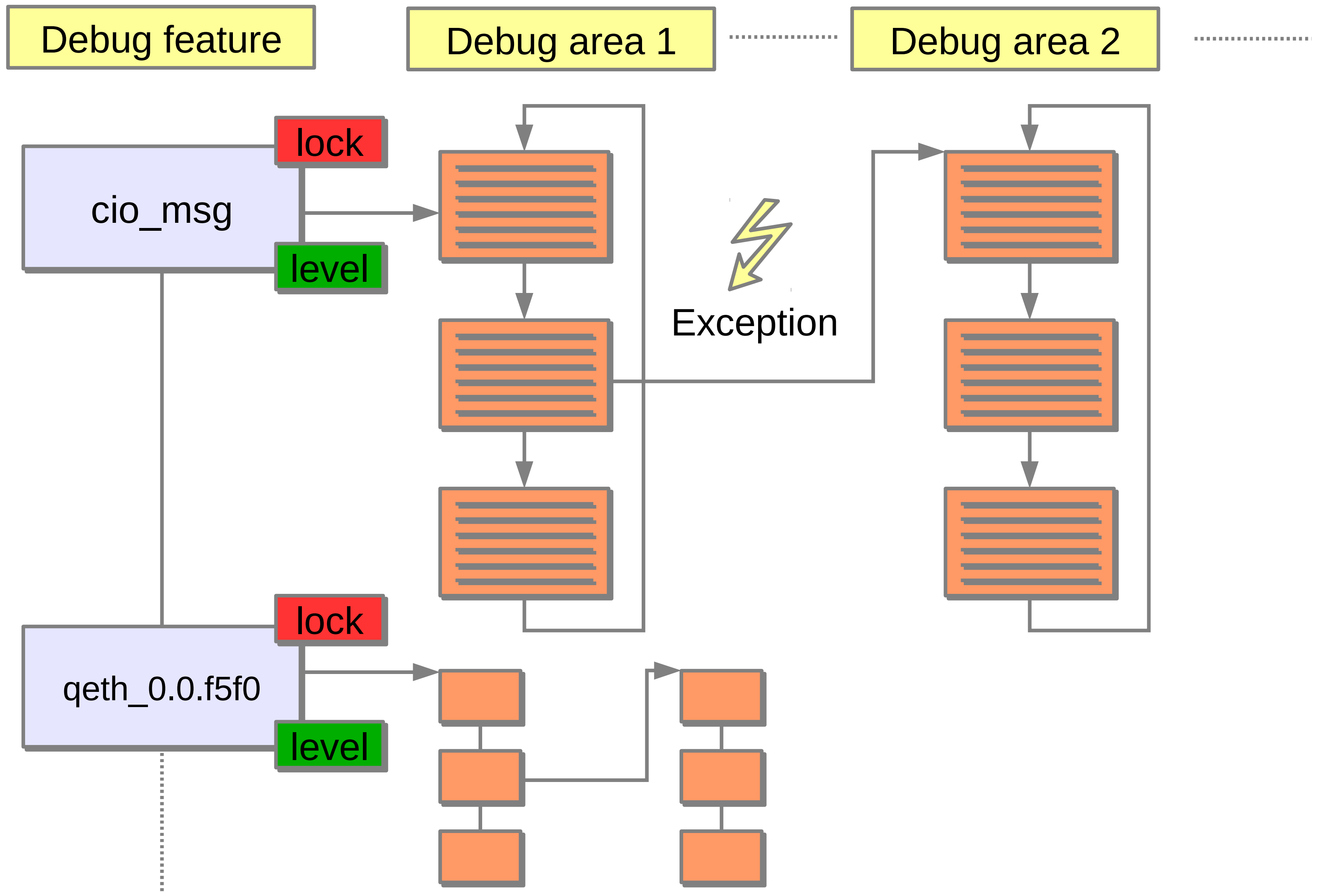
Introduction

S390 debug feature

Tracepoints

Comparison & Outlook

s390dbf - Design



s390dbf - Design

- **Used by s390 device drivers**
- **All debug entries of one debug feature have fixed size**
- **Debug entry metadata:**
  - Timestamp
  - CPU-Number of calling task
  - Debug level of debug entry (0...6)
  - Return address to caller
  - Exception flag
- **Events are logged when “entry level” <= “debug level”**
- **Two formatter views:**
  - debug_hex_ascii_view
  - debug_sprintf_view
- **The sprintf view only stores “unsigned longs”**
  - Formatting when reading view

s390dbf - API

- Create debug feature:

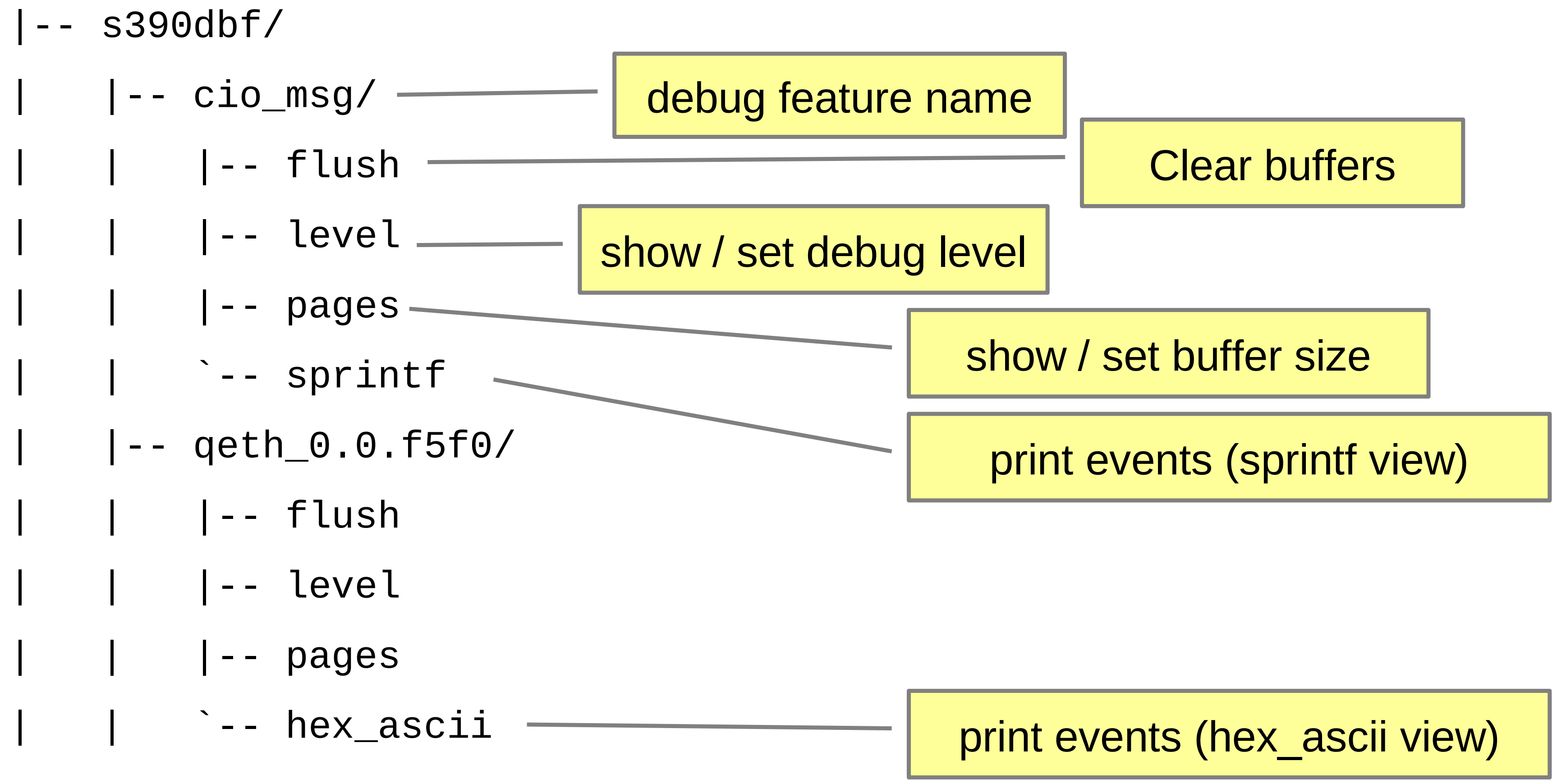
  ```
  – debug_info_t *debug_register(char *name, int pages, int areas, int size);
  ```

- Write events:

  ```
  – debug_entry_t *debug_event(debug_info_t *id, int level, void *data,
                               int length);
  – debug_entry_t *debug_sprintf_event(debug_info_t *id, int level,
                                       char *fmt,...);
  ```

- Documentation/s390/s390dbf.txt

s390dbf - debugfs: /sys/kernel/debug/s390dbf/



```
|-- s390dbf/
|   |-- cio_msg/
|   |   |-- flush
|   |   |-- level
|   |   |-- pages
|   |   `-- sprintf
|   |-- qeth_0.0.f5f0/
|   |   |-- flush
|   |   |-- level
|   |   |-- pages
|   |   `-- hex_ascii
```

s390dbf - debugfs: Print views

```
# cat cio_msg/sprintf
00 01401200730:177434 2 - 01 5b1fea  snsid: device 0.0.379b: rc=0 3990/e9 3390/0c
00 01401200730:177434 2 - 00 5ad836  event: sch 0.0.0010, process=1, action=2
00 01401200730:177448 2 - 00 5b1fea  snsid: device 0.0.000e: rc=0 1403/00 0000/00 (diag210)
00 01401200730:177455 2 - 00 5ad836  event: sch 0.0.0011, process=1, action=2
00 01401200730:177458 2 - 00 5b1fea  snsid: device 0.0.379a: rc=0 3990/e9 3390/0c
00 01401200730:177498 2 - 00 5ad836  event: sch 0.0.0013, process=1, action=2
00 01401200730:177621 2 - 01 5b1fea  snsid: device 0.0.1703: rc=0 1731/03 1732/03
00 01401200730:177674 2 - 00 5b1fea  snsid: device 0.0.0190: rc=0 3990/e9 3390/0c
00 01401200730:177839 2 - 00 5b1fea  snsid: device 0.0.019d: rc=0 3990/e9 3390/0c

# cat qeth_0.0.f5f0/hex_ascii
00 01401200796:957057 5 - 00 606a92  73 6b 62 72 00 00 00 00 | skbr....
00 01401200796:957058 5 - 00 60694a  32 31 31 33 61 35 30 30 | 2113a500
00 01401200796:957186 6 - 00 61095e  66 69 6c 6c 62 66 6e 70 | fillbfnp
00 01401200796:957213 6 - 00 6103fa  6e 70 2d 3e 70 61 63 6b | np->pack
00 01401200796:957312 5 - 01 606a92  73 6b 62 72 00 00 00 00 | skbr....
```

s390dbf - Dump support

```
# crash vmlinux dump.elf
      KERNEL: vmlinux
    DUMPFILE: dump.elf
...

# crash> s390dbf
Debug Logs:
==================
 - cio_msg
 - qeth_0.0.f5f0
 - qdio_0.0.f505

# crash> s390dbf cio_msg sprintf
0 1400853628:378269 2 - 00 <io_sch_event+1490> event: sch 0.0.0007, process=1
0 1400853628:378286 2 - 01 <verify_done+0250>  vrfy : device 0.0.0009: rc=0
0 1400853628:378633 2 - 00 <io_sch_event+1188> event: sch 0.0.0000, action=2
```

Tracepoints - Overview

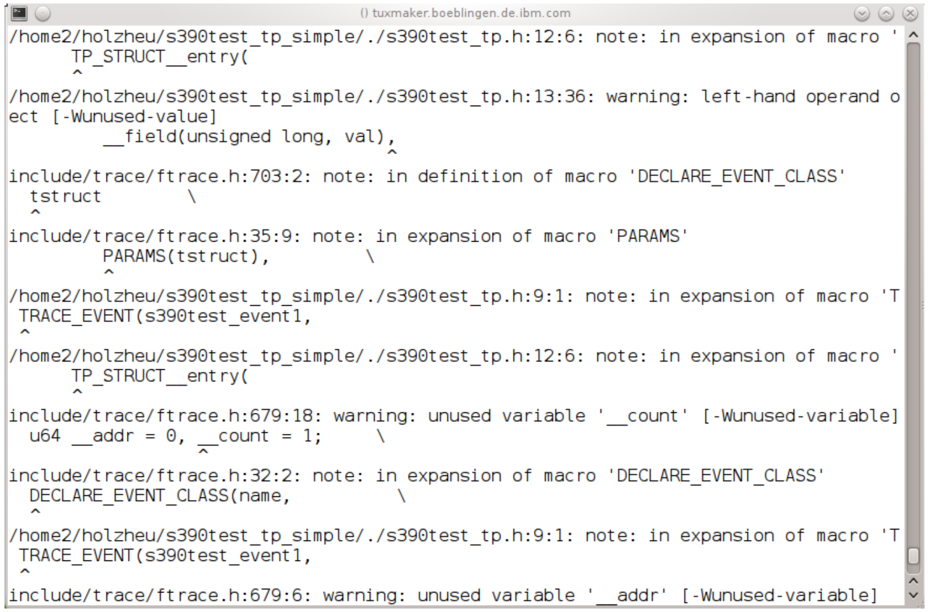
- **Maintainer: Steven Rostedt (Red Hat)**
- **Two level hierarchy:**
  - Trace systems
  - Events (unique)
- **About 35 trace systems with 1000 tracepoints in current s390 kernel**
  - Scheduler
  - System calls
  - Block layer
  - SCSI
  - Network
  - File systems (EXT, XFS), ...

Tracepoints - Overview

- **Per-CPU Ring Buffers**
  - One main buffer
  - Per-Instance buffers
  - Snapshot buffers
- **Uses Jump-labels**
  - NOP for disabled events

Tracepoints - API

- **Trace macros in `include/linux/tracepoints.h`**
  - `TRACE_EVENT()`
  - `DECLARE_EVENT_CLASS()/DEFINE_EVENT()`
- **Macros generate trace code**
- **Note: Sometimes hard to debug errors in macro definition**

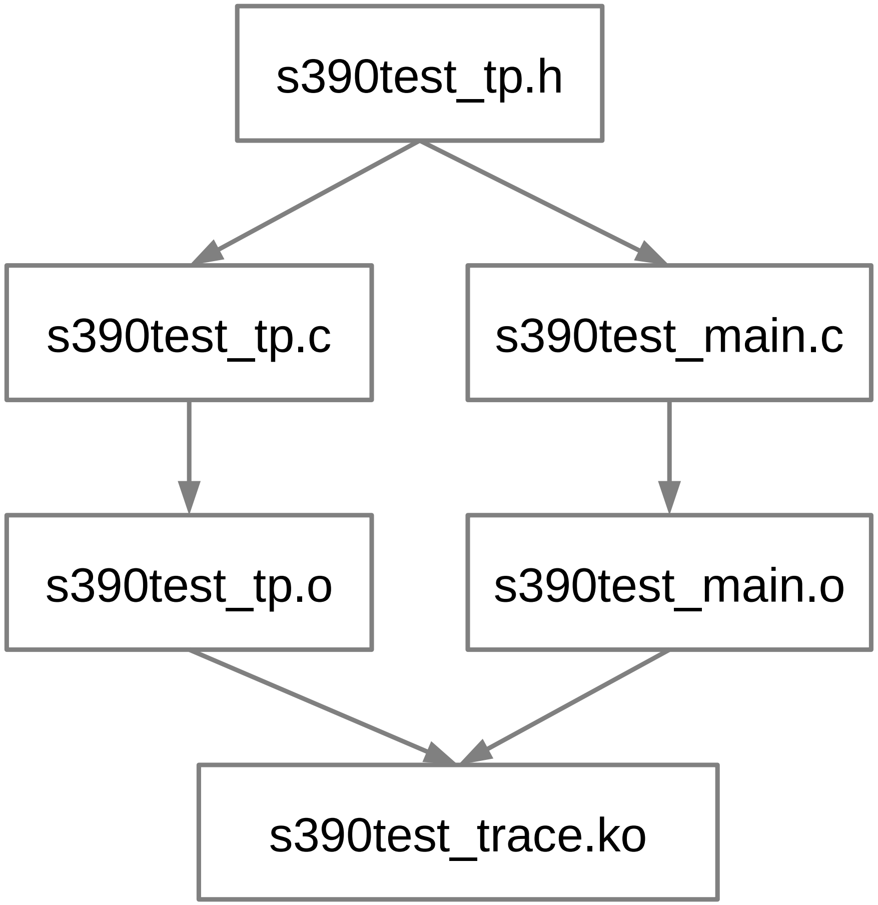


() tuxmaker.boeblingen.de.ibm.com

```
/home2/holzheu/s390test_tp_simple/./s390test_tp.h:12:6: note: in expansion of macro '
      TP_STRUCT__entry(
      ^
/home2/holzheu/s390test_tp_simple/./s390test_tp.h:13:36: warning: left-hand operand o
ect [-Wunused-value]
         __field(unsigned long, val),
                                    ^
include/trace/ftrace.h:703:2: note: in definition of macro 'DECLARE_EVENT_CLASS'
  tstruct        \
  ^
include/trace/ftrace.h:35:9: note: in expansion of macro 'PARAMS'
         PARAMS(tstruct),         \
         ^
/home2/holzheu/s390test_tp_simple/./s390test_tp.h:9:1: note: in expansion of macro 'T
 TRACE_EVENT(s390test_event1,
 ^
/home2/holzheu/s390test_tp_simple/./s390test_tp.h:12:6: note: in expansion of macro '
      TP_STRUCT__entry(
      ^
include/trace/ftrace.h:679:18: warning: unused variable '__count' [-Wunused-variable]
  u64 __addr = 0, __count = 1;     \
                  ^
include/trace/ftrace.h:32:2: note: in expansion of macro 'DECLARE_EVENT_CLASS'
  DECLARE_EVENT_CLASS(name,           \
  ^
/home2/holzheu/s390test_tp_simple/./s390test_tp.h:9:1: note: in expansion of macro 'T
 TRACE_EVENT(s390test_event1,
 ^
include/trace/ftrace.h:679:6: warning: unused variable '__addr' [-Wunused-variable]
```

Tracepoints - API: Example

- **Trace System:** `s390test`
- **Trace Event:** `s390test_event1`
- **Kernel module:** `s390test_trace.ko`
  - Define TP:
    - `s390test_tp.h`
  - Create TP definitions:
    - `s390test_tp.c`
  - Trigger TP:
    - `s390test_main.c`



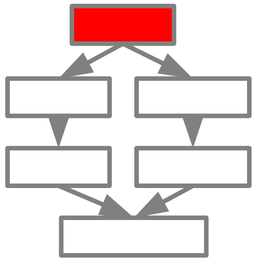
Tracepoints - API: TRACE_EVENT (s390test_tp.h)

- **TRACE_SYSTEM**: Trace system name
- **TP_PROTO/TP_ARGS**: Signature
- **TP_STRUCT_entry**: Payload
- **TP_fast_assign**: Copy entries (code)
- **TP_printk**: Format entry (code)

```
#undef TRACE_SYSTEM
#define TRACE_SYSTEM s390test
#include <linux/tracepoint.h>

TRACE_EVENT(s390test_event1,
     TP_PROTO(unsigned long _val),
     TP_ARGS(_val),
     TP_STRUCT__entry(
          __field(unsigned long, val)
                     ),
     TP_fast_assign(
          __entry->val = _val;
                    ),
     TP_printk("val=%lu", __entry->val)
);
```

<system>_<event>

sorry, macro magic

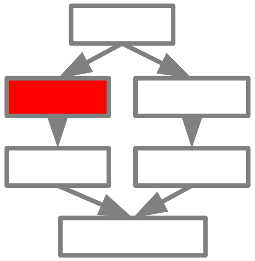


Tracepoints - API: Define Trace (s390test_tp.c)

- **CREATE_TRACE_POINTS**
  - Macro expansion
  - Create code and data
  - Only in *one* source file

```
#define CREATE_TRACE_POINTS

#include "s390test_tp.h"
```

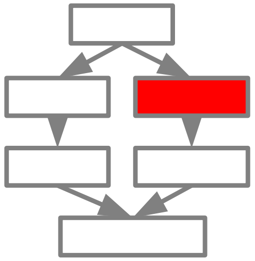
```
struct tracepoint __tracepoint_s390test_event1
__attribute__((section("__tracepoints"))) = {

     __tpstrtab_s390test_event1,

     ((struct static_key) { .enabled = { (0) } }),

     ((void *)0),

     ((void *)0),

     ((void *)0)

};

.... <lots of other stuff>
```



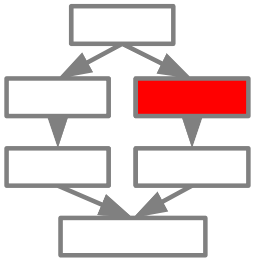
Tracepoints - API: Trigger tracepoint (s390test_main.c)

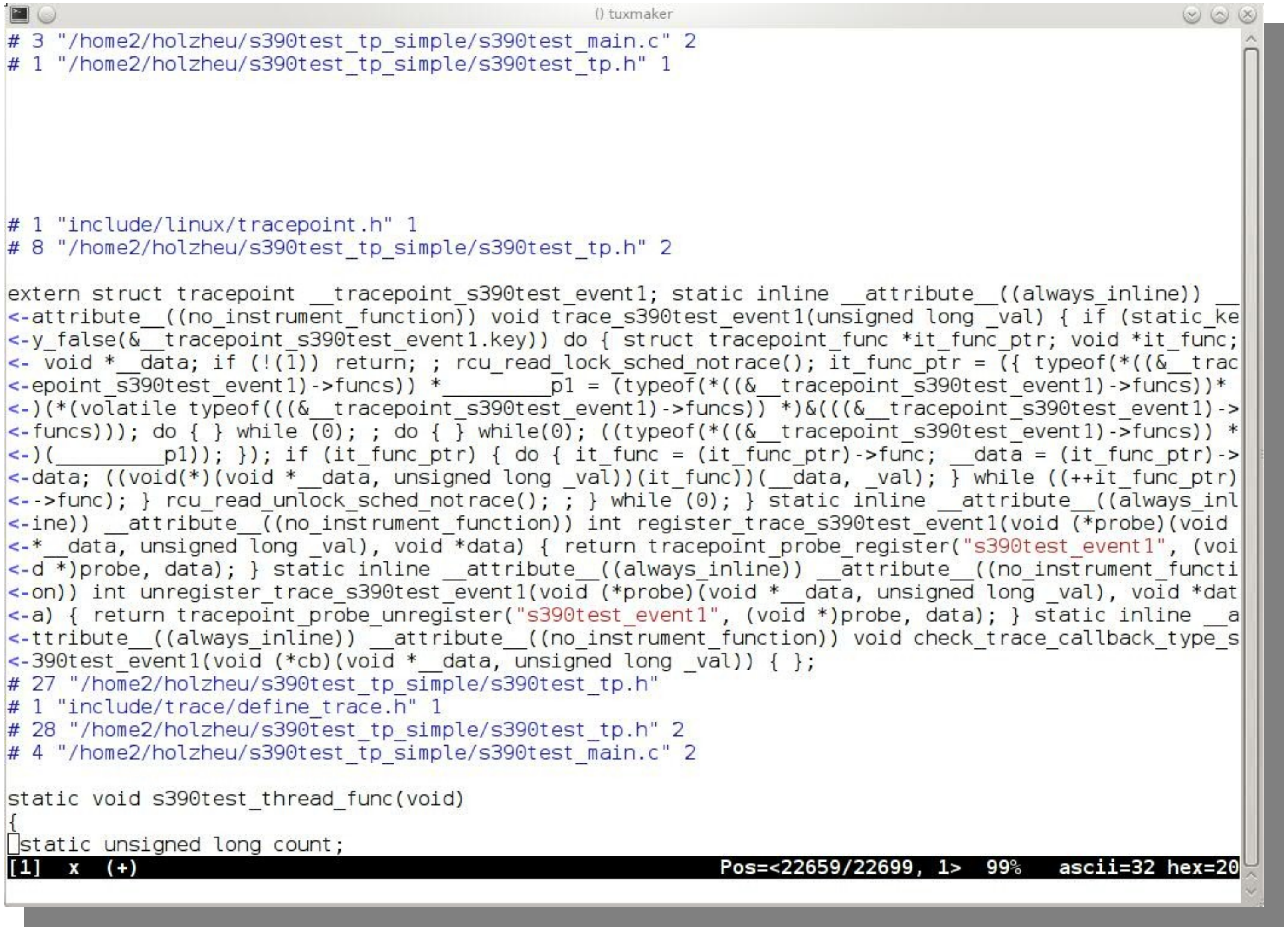
- Function call: `trace_<tp name>(...)`
- Inline function created by `TRACE_EVENT` macro

```
#include "s390test_tp.h"


static void
s390test_thread_func(void)
{
    unsigned long i;
    for(i = 1; i <= 2; i++)
        trace_s390test_event1(i);
}
```



Tracepoints - API: Define Trace (s390test_main.c macro expansion)

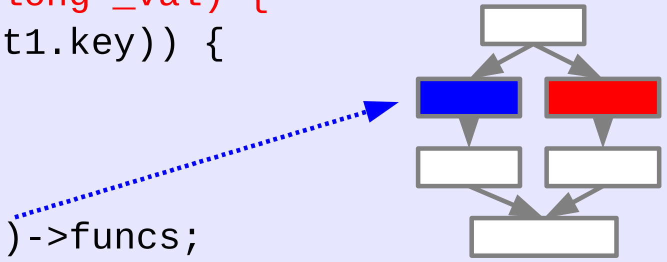




```
# 3 "/home2/holzheu/s390test_tp_simple/s390test_main.c" 2
# 1 "/home2/holzheu/s390test_tp_simple/s390test_tp.h" 1






# 1 "include/linux/tracepoint.h" 1
# 8 "/home2/holzheu/s390test_tp_simple/s390test_tp.h" 2

extern struct tracepoint __tracepoint_s390test_event1; static inline __attribute__((always_inline)) __
<-attribute__((no_instrument_function)) void trace_s390test_event1(unsigned long _val) { if (static_ke
<-y_false(&__tracepoint_s390test_event1.key)) do { struct tracepoint_func *it_func_ptr; void *it_func;
<- void *__data; if (!(1)) return; ; rcu_read_lock_sched_notrace(); it_func_ptr = ({ typeof(*((&__trac
<-epoint_s390test_event1)->funcs)) *__________p1 = (typeof(*((&__tracepoint_s390test_event1)->funcs))*
<-)(*(volatile typeof(((&__tracepoint_s390test_event1)->funcs)) *)&(((&__tracepoint_s390test_event1)->
<-funcs))); do { } while (0); ; do { } while(0); ((typeof(*((&__tracepoint_s390test_event1)->funcs)) *
<-)(__________p1)); }); if (it_func_ptr) { do { it_func = (it_func_ptr)->func; __data = (it_func_ptr)->
<-data; ((void(*)(void *__data, unsigned long _val))(it_func))(__data, _val); } while ((++it_func_ptr)
<-->func); } rcu_read_unlock_sched_notrace(); ; } while (0); } static inline __attribute__((always_inl
<-ine)) __attribute__((no_instrument_function)) int register_trace_s390test_event1(void (*probe)(void
<-*__data, unsigned long _val), void *data) { return tracepoint_probe_register("s390test_event1", (voi
<-d *)probe, data); } static inline __attribute__((always_inline)) __attribute__((no_instrument_functi
<-on)) int unregister_trace_s390test_event1(void (*probe)(void *__data, unsigned long _val), void *dat
<-a) { return tracepoint_probe_unregister("s390test_event1", (void *)probe, data); } static inline __a
<-ttribute__((always_inline)) __attribute__((no_instrument_function)) void check_trace_callback_type_s
<-390test_event1(void (*cb)(void *__data, unsigned long _val)) { };
# 27 "/home2/holzheu/s390test_tp_simple/s390test_tp.h"
# 1 "include/trace/define_trace.h" 1
# 28 "/home2/holzheu/s390test_tp_simple/s390test_tp.h" 2
# 4 "/home2/holzheu/s390test_tp_simple/s390test_main.c" 2

static void s390test_thread_func(void)
{
 static unsigned long count;
[1]  x  (+)                                        Pos=<22659/22699, 1>  99%   ascii=32 hex=20
```

Tracepoints - API: Define Trace (s390test_main.c macro expansion)

```
static inline void trace_s390test_event1(unsigned long _val) {
  if (static_key_false(&__tracepoint_s390test_event1.key)) {
       struct tracepoint_func *it_func_ptr;
       void *it_func;
       void *__data
       it_func_ptr = &__tracepoint_s390test_event1)->funcs;
       if (it_func_ptr) {
       do {
         it_func = (it_func_ptr)->func;
         __data = (it_func_ptr)->data;
         ((void(*)(void *__data, unsigned long _val))it_func)(__data, _val);
       } while ((++it_func_ptr)->func);
  }
}
static inline int
register_trace_s390test_event1(void (*probe)(void *__data, unsigned long _val),
                                void *data) {
        return tracepoint_probe_register("s390test_event1", (void *)probe,
                                          data);
}
```



Tracepoints - API: Field Types

- **`TP_STRUCT_entry()`**
  - `__field(type, name)`: Simple field in the structure
  - `__array(type, name, len)` : Defines an array
  - `__string(name, source)` : Variable length null terminated string
  - `__dynamic_array(type, item, len)`: Variable length array where *len* is a variable
- **`TP_fast_assign()`**
  - `__assign_str(name, source)` : Assign variable length null terminated strings
- **`TP_printk()`**
  - `__print_hex(__entry->buffer, len)` : Print a hex dump
  - `__print_flags(flags, delimiter, values)` : Print symbolic names for flags
  - `__print_symbolic(val, values)` : Print symbolic names for exact matches

Tracepoints - API: Define custom formatting function

```
const char *debug_trace_hex_ascii_seq(struct trace_seq *p,
                                      const unsigned char *buf, int buf_len)
{
       const char *ret = p->buffer + p->len;
       int i;
       for (i = 0; i < buf_len; i++)
               trace_seq_printf(p, "%s%2.2x", i == 0 ? "" : " ", buf[i]);
       for (i = 0; i < buf_len; i++)
           trace_seq_printf(p, "%c", isascii(buf[i]) && isprint(buf[i]) ? buf[i] : '.');
       trace_seq_putc(p, 0);
       return ret;
}

#define __print_hex_ascii(buf, len) debug_trace_hex_ascii_seq(p , buf, len)
```

- Define function that returns a static string
- `trace_seq_printf()` and `trace_seq_put()` can be used

Tracepoints - API: Define custom formatting function

```
TRACE_EVENT(s390dbf_event,
       TP_PROTO(debug_info_t *id, void *buf, int len),
       TP_ARGS(id, buf, len),
       TP_STRUCT__entry(
               __field(int, len)
               __dynamic_array(char, name, 24)
               __dynamic_array(u8, buf, len)
       ),
       TP_fast_assign(
               __entry->len = len;
               strncpy(__get_dynamic_array(name), id->name, 23);
               memcpy(__get_dynamic_array(buf), buf, len);
       ),
       TP_printk("%s: %s", (char *)__get_str(name),
                 __print_hex_ascii(__get_dynamic_array(buf), __entry->len)
       )
```

- Function can be used like `__print_hex()` in `TP_printk`

Tracepoints - API: Connect a probe to a tracepoint

- Connect a function (probe) to a tracepoint
- Multiple probes are possible
- register_trace_<tp name>(fn, data)
- Register function created by Macro magic

```
TRACE_EVENT(sched_sw,
        TP_PROTO(struct task_struct *prev,
                 struct task_struct *next),
 ...
 );

static void probe_sched_sw(void *ignore,
        struct task_struct *prev,
        struct task_struct *next)
{...}

register_trace_sched_switch(probe_sched_sw,
                            NULL);
```

Macro Expansion:

```
tracepoint_probe_register("sched_sw", probe,
                           data);
```

Tracepoints - API: DECLARE_EVENT_CLASS

- **Two macros**
  - `DECLARE_EVENT_CLASS`: Define template
  - `DEFINE_EVENT`: Define event of specified class
- **For multiple events with same signature**
- **Saves**
  - Lines of code
  - Static memory (will show later)

Tracepoints - API: DECLARE_EVENT_CLASS

```
DECLARE_EVENT_CLASS(s390test_class,
     TP_PROTO(unsigned long _val),
     TP_ARGS(_val),
     TP_STRUCT__entry(
          __field(unsigned long, val)
                     ),
     TP_fast_assign(
          __entry->val = _val;
                    ),
     TP_printk("val=%lu", __entry->val)
);
DEFINE_EVENT(s390test_class, s390test_event1,
     TP_PROTO(unsigned long _val),
     TP_ARGS(_val)
);
DEFINE_EVENT(s390test_class, s390test_event2,
     TP_PROTO(unsigned long _val),
     TP_ARGS(_val)
);
```

Redundancy necessary because of macro magic

Tracepoints - Debugfs: /sys/kernel/debug/tracing

```
|-- README                          HOWTO
|-- trace                           get formatted trace buffer
|-- trace_pipe                      stream trace buffer (consuming)
|...
|-- events/
|   |-- enable                      enable all events
|   |-- block/
|   |   |-- enable                  enable all block events
|   |   |-- block_bio_backmerge
|   |   |   |-- enable              enable block_bio_backmerge event
|...
|-- instances/                      create / remove instances
                                    mkdir / rmdir
|-- per_cpu/                        per-CPU data
|   |-- cpu0/
|   |   |-- trace_pipe
```

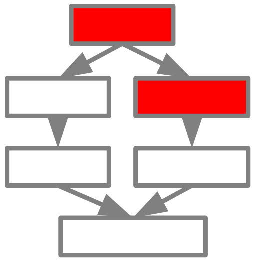
Example: s390test_trace.ko

```
#undef TRACE_SYSTEM
#define TRACE_SYSTEM s390test
#include <linux/tracepoint.h>

TRACE_EVENT(s390test_event1,
     TP_PROTO(unsigned long _val),
     TP_ARGS(_val),
     TP_STRUCT__entry(
          __field(unsigned long, val)
                     ),
     TP_fast_assign(
          __entry->val = _val;
                    ),
     TP_printk("val=%lu", __entry->val)
);
```

s390test_tp.h

```
#include "s390test_tp.h"


static void s390test_thread_func(void)
{
    unsigned long i;
    for(i = 1; i <= 2; i++)
           trace_s390test_event1(i);
}
```

s390test_main.c



Tracepoints - Debugfs: Basics

- Load module
- Enter trace directory
- Enable event (per-event)
- Read trace buffer
- Read trace pipe

```
# insmod ./s390test_trace.ko
# cd /sys/kernel/debug/tracing/
# echo 1 > events/s390test/s390test_event1/enable
# cat trace
 s390test-704[000] 102540.659351: s390test_event1: val=1
 s390test-704[000] 102540.659354: s390test_event1: val=2
# cat trace_pipe
 s390test-704[000] 102540.659351: s390test_event1: val=1
 s390test-704[000] 102540.659354: s390test_event1: val=2
```

Tracepoints - Debugfs: Basics 2

- Query total buffer size
- Set (per-CPU) buffer size
- See per-CPU info

```
# cat buffer_total_size_kb
448 (expanded: 90112)
# echo 1024 > buffer_size_kb
# ls per_cpu/
cpu0   cpu14  cpu2 ...
# ls per_cpu/cpu0/
buffer_size_kb  snapshot_raw  trace       trace_pipe_raw
snapshot        stats         trace_pipe
```

Tracepoints - Debugfs: Instances

- Create instance
- Enable event
- Display trace

```
# cd instances
# mkdir myinstance
# cd myinstance
# echo 1 > events/s390test/enable
# cat trace
s390test704 [000] 105117.659315: s390test_event1: val=1
s390test704 [000] 105117.659315: s390test_event1: val=2
```

Tracepoints - Debugfs: Snapshots

- Create snapshot
- Display snapshot
- Free snapshot

```
# echo 1 > snapshot
# cat snapshot
s390test704  [000] 105117.659315: s390test_event1: val=1
s390test704  [000] 105117.659315: s390test_event1: val=2
# echo 0 > snapshot
```

Tracepoints - Debugfs: Event filter

- Show global filter help
- Enable local filter
- Boolean operations
- Predicate Tree

```
# cat events/s390test/filter
  ### global filter ###
  # Use this to set filters for multiple events.
  # Only events with the given fields will be affected.
  # If no events are modified, an error message will be
  # displayed here

# echo "val==2" > events/s390test/s390test_event1/filter
# cat trace
s390test-704 [000] 105733.659376: s390test_event1: val=2
s390test-704 [000] 105734.659385: s390test_event1: val=2

# echo "val==1||val==2" > \
        events/s390test/s390test_event1/filter
s390test-704 [000] 105735.659376: s390test_event1: val=1
s390test-704 [000] 105736.659385: s390test_event1: val=2
```

Tracepoints - Debugfs: Event trigger

- Set trigger
- Use boolean operation
- Disable trigger
- Supported triggers:
  - enable_event
  - disable_event
  - stacktrace
  - traceon
  - traceoff

```
# echo 'stacktrace if val==2' > s390test_event1/trigger
# cat trace
s390test-739 [000] 745.949957: s390test_event1: val=1
s390test-739 [000] 745.949958: s390test_event1: val=2
s390test-739 [000] 745.949959: <stack trace>
 => kthread
 => kernel_thread_starter
s390test-739 [000] 745.949959: s390test_event1: val=3

# echo 'stacktrace if val==2 || val=3 ' > \
          s390test_event1/trigger

# echo '!stacktrace if val==2 || val=3 ' > \
          s390test_event1/trigger
```

Tracepoints - Kernel parameter

- **trace_event=[event-list]**
  - Set and start specified trace events in order to facilitate *early boot* debugging.
  - Also wild cards can be used:
    - The buf format can be <subsystem>:<event-name>
    - <event-name> or :<event-name> means any event by that name.
    - <subsystem>:* or <subsystem>: means all events in that subsystem
    - <name> (no ':') means all events in a subsystem with the name <name> or any event that matches <name>

- **Example**
  - trace_event=s390test:

Tracepoints - traceevent library

- In kernel code: tools/lib/traceevent
- Currently copied into tools (e.g. trace-cmd, RAS daemon)
- Provides similar interface as kernel

```
– int pevent_get_field_val(struct trace_seq *s, struct event_format *event,
                           const char *name, struct pevent_record *record,
                           unsigned long long *val, int err);

– int trace_seq_printf(struct trace_seq *s, const char *fmt, ...);
```

Tracepoints - trace-cmd

- **Plugins for event formatting**
- **Commands:**
  - record: Record a trace into a file trace.dat
    - `trace-cmd record -e syscalls:sys_enter_write`
  - report: Read out the trace stored in the trace.dat file
    - `trace-cmd report`
  - start: Start tracing without recording into a file
    - `trace-cmd start -e syscalls:sys_enter_write`
  - stop: Stop the kernel from recording trace data
    - `trace-cmd stop -e syscalls:sys_enter_write`

Tracepoints - perf command

- Get counter of sys_enter_write trace point for dd process
- Get global counter of `sys_enter_write` trace point

```
# perf stat -e syscalls:sys_enter_write \
              dd if=/dev/zero of=out  count=100
100+0 records in
100+0 records out
51200 bytes (51 kB) copied, 0.000300031 s, 171 MB/s

Performance counter stats for
       'dd if=/dev/zero of=out count=100':
               103      syscalls:sys_enter_write
       0.001036218 seconds time elapsed


# perf stat -a -e syscalls:sys_enter_write sleep 10
 Performance counter stats for 'system wide':
             10024      syscalls:sys_enter_write
      10.000604627 seconds time elapsed
```

Tracepoints - perf command (record/report)

- Get more details for sys_enter_read trace point with the "*perf record*" and "*perf report*" commands

```
# perf record -a -e syscalls:sys_enter_read sleep 10
 Performance counter stats for 'system wide':
            10024      syscalls:sys_enter_write
     10.000604627 seconds time elapsed

# perf report
Samples: 5K of event 'syscalls:sys_enter_read', Event
count (approx.): 5030
 99.40%        dd  libc-2.15.so        __GI___libc_read
  0.20%  rsyslogd  libpthread-2.15.so  x0000000000011f98
  0.12%      sshd  libc-2.15.so        __GI___libc_read
  0.10%      bash  libc-2.15.so        __GI___libc_read
  0.06%     crond  libc-2.15.so        __GI___libc_read
  0.06%        dd  ld-2.15.so           read
  0.04%  sendmail  libc-2.15.so        __GI___libc_read
  0.02%     sleep  ld-2.15.so          read
```

Tracepoints - crash plugin (trace.so)

- Crash extension: trace.so
- Can call trace-cmd under the covers
- Can dump the (main) trace buffers into file
- Then trace-cmd is used to format trace

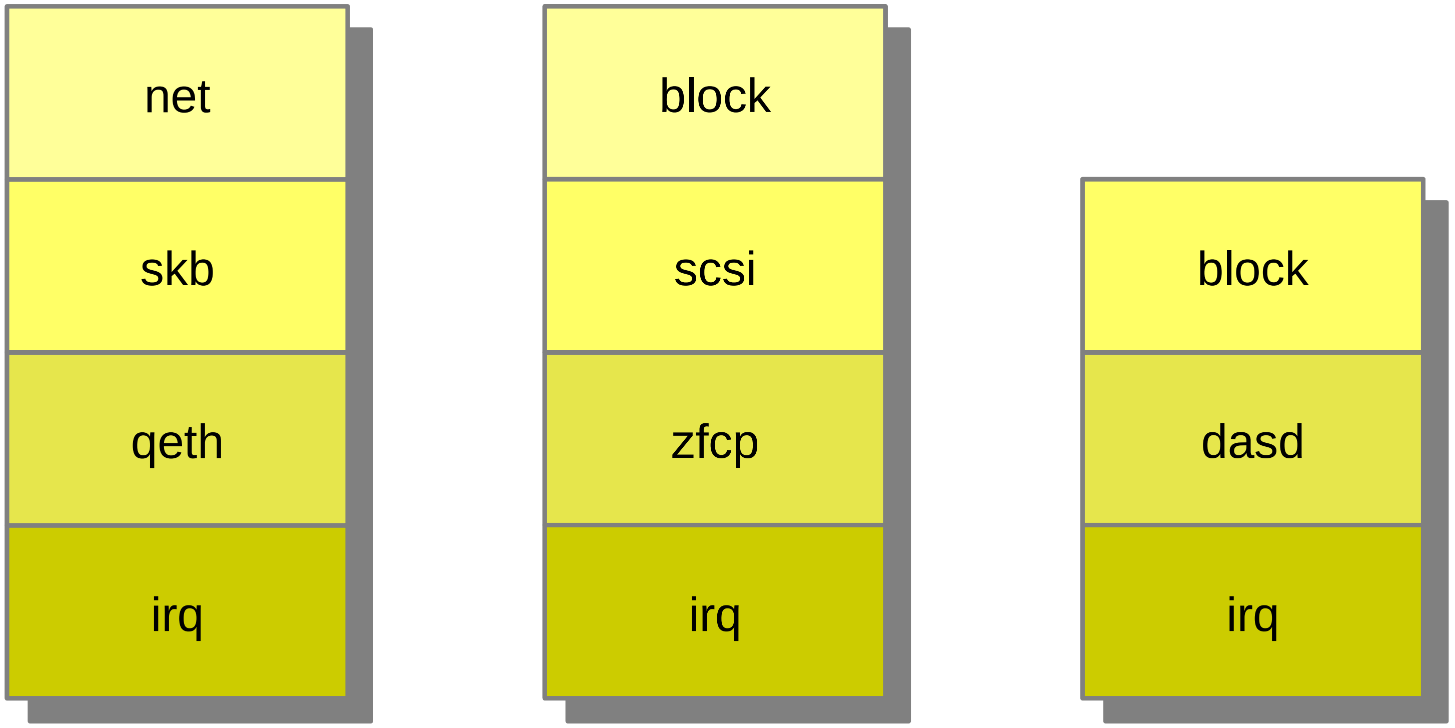
```
# crash dump vmlinux
crash> extend trace.so
crash> trace dump -t
crash> exit

# trace-cmd report trace.dat
s390test-704 [000] 105735.659376: s390test_event1: val=1
s390test-704 [000] 105736.659385: s390test_event1: val=2
```

Tracepoints - Combine Function tracer with tracepoints

```
# cd /sys/kernel/debug/tracing/
# echo "do_IRQ generic_handle_irq qeth_qdio_output_handler" > set_ftrace_filter
# echo function_graph > current_tracer
# echo 1 > events/net/netif_receive_skb/enable
# cat trace_pipe
  0)                |  do_IRQ() {
  0)   1.949 us     |    generic_handle_irq();
  0)                |    qeth_qdio_output_handler() {
  0)                |  /* netif_receive_skb: dev=eth0 skbaddr=31589700 len=52 */
  0) + 12.555 us    |    }
  0) + 32.872 us    |  }
```

Tracepoints - Combine different trace systems

- There are several subsystems within the kernel stack that have a relation
- Tracepoints can show processing through the stack



Tracepoints - Combine different trace systems (network)

```
# echo 1 > irq/enable

# echo "name==AIO" > irq/filter  # only trace async interrupts

# echo 1 > skb/enable

# echo 1 > net/enable

# cat ../trace_pipe
```

Tracepoints - Combine different trace systems (network recv)

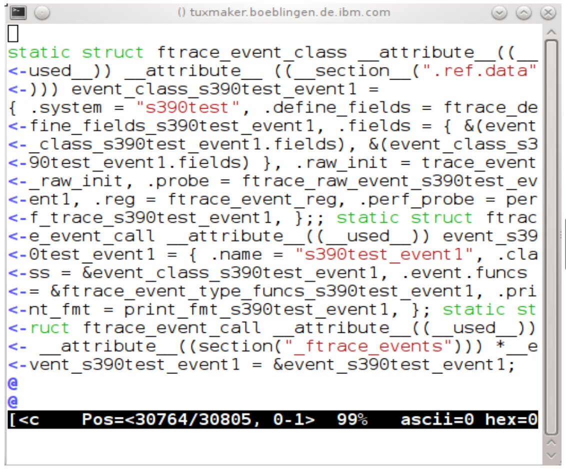
```
/* Inbound data available */
<idle>-0 1366.023268: irq_handler_entry: irq=3 name=AIO
/* Start polling - napi schedule  */
<idle>-0 1366.023270: softirq_raise: vec=3 [action=NET_RX]
<idle>-0 1366.023270: irq_handler_exit: irq=3 ret=handled
<idle>-0 1366.023271: softirq_entry: vec=3 [action=NET_RX]
/* qeth polling function - invokes napi_gro_receive */
<idle>-0 1366.023277: napi_gro_receive_entry: dev=eth0 napi_id=0x0
                      queue_mapping=0 skbaddr=000000002f15f700 vlan_tagged=0
                      vlan_proto=0x0000 vlan_tci=0x0000 protocol=0x0800 ip_summed=1
                      rxhash=0x00000000 l4_rxhash=0 len=100 data_len=0 truesize=1024
                      mac_header_valid=1 mac_header=-14 nr_frags=0 gso_size=0
/* Process receive buffer from network */
<idle>-0 1366.023280: netif_receive_skb: dev=eth0 skbaddr=000000002f15f700 len=100
<idle>-0 1366.023292: softirq_exit: vec=3 [action=NET_RX]
/* Copy received data to user space */
sshd-681 1366.023309: skb_copy_datagram_iovec: skbaddr=000000002f15f700 len=48
```

Tracepoints - Static memory usage (Kernel 3.14 / s390-64bit)

- TRACE_EVENT: **1.292 bytes**
- DECLARE_EVENT_CLASS
  - DECLARE_EVENT_CLASS: **952 bytes**
  - DEFINE_EVENT: **340 bytes**

TRACE_EVENT(s390test_event1)
TRACE_EVENT(s390test_event2)
TRACE_EVENT(s390test_event3)

DECLARE_EVENT_CLASS(class)
DEFINE_EVENT(class, s390test_event1)
DEFINE_EVENT(class, s390test_event2)
DEFINE_EVENT(class, s390test_event3)



```
static struct ftrace_event_class __attribute__((__
<-used__)) __attribute__ ((__section__(".ref.data"
<-))) event_class_s390test_event1 =
{ .system = "s390test", .define_fields = ftrace_de
<-fine_fields_s390test_event1, .fields = { &(event
<-_class_s390test_event1.fields), &(event_class_s3
<-90test_event1.fields) }, .raw_init = trace_event
<-_raw_init, .probe = ftrace_raw_event_s390test_ev
<-ent1, .reg = ftrace_event_reg, .perf_probe = per
<-f_trace_s390test_event1, };; static struct ftrac
<-e_event_call __attribute__((__used__)) event_s39
<-0test_event1 = { .name = "s390test_event1", .cla
<-ss = &event_class_s390test_event1, .event.funcs
<-= &ftrace_event_type_funcs_s390test_event1, .pri
<-nt_fmt = print_fmt_s390test_event1, }; static st
<-ruct ftrace_event_call __attribute__((__used__))
<- __attribute__((section("_ftrace_events"))) *__e
<-vent_s390test_event1 = &event_s390test_event1;
@
@
[<c    Pos=<30764/30805, 0-1>  99%   ascii=0 hex=0
```

Section	1	2	3	1C	2C	3C	3-2	3C - 2C
.text	884	1556	2228	884	952	1020	672	68
.init.text	0	0	0	0	0	0	0	0
.exit.text	0	0	0	0	0	0	0	0
.rodata.str1.2	86	102	118	86	102	118	16	16
.rodata	28	48	68	28	28	28	20	0
.modinfo	0	0	0	0	0	0	0	0
__tracepoints_strings	16	32	48	16	32	48	16	16
.eh_frame	392	608	832	392	392	392	224	0
__versions	1408	1408	1408	1408	1408	1408	0	0
.data	8	8	8	8	8	8	0	0
__jump_table	24	48	72	24	48	72	24	24
.data.rel.ro.local	0	0	0	0	0	0	0	0
_ftrace_events	8	16	24	8	16	24	8	8
.data.rel.local	176	352	528	176	320	464	176	144
.ref.data	72	144	216	72	72	72	72	0
__tracepoints_ptrs	8	16	24	8	16	24	8	8
__tracepoints	56	112	168	56	112	168	56	56
.gnu.linkonce.this_module	648	648	648	648	648	648	0	0
.bss	16	16	16	16	16	16	0	0
Memory in bytes	**3830**	**5114**	**6406**	**3830**	**4170**	**4510**	**1292**	**340**

Comparison - Trace buffer memory consumption

- **Tracepoints: per-CPU buffer for each online CPU**
- **s390dbf: One global buffer per debug feature**
- **“Tracepoints buffer size” = “s390dbf buffer size” / “online CPU count” ?**
  - Probably not
  - TP will consume more memory

Comparison - Performance

tracepoints

```
TRACE_EVENT(s390test_event1,
     TP_PROTO(unsigned long _val),
     TP_ARGS(_val),
     TP_STRUCT__entry(
          __field(unsigned long, val)
                     ),
     TP_fast_assign(
          __entry->val = _val;
                    ),
     TP_printk("val=%lu", __entry->val)
);

static void tp_int(void)
{
   unsigned long i;
   for (i = 0; i < loops; i++)
        trace_s390test_event1(i);
}
```

s390dbf

```
static void dbf_int(void)
{
  debug_info_t *dbf;
  unsigned long i;
  dbf = debug_register("s390test", 1,
                      1, sizeof(long));
  for (i = 0; i < loops; i++)
    debug_event(dbf, 2, &i, sizeof(i));
  ...
}
```

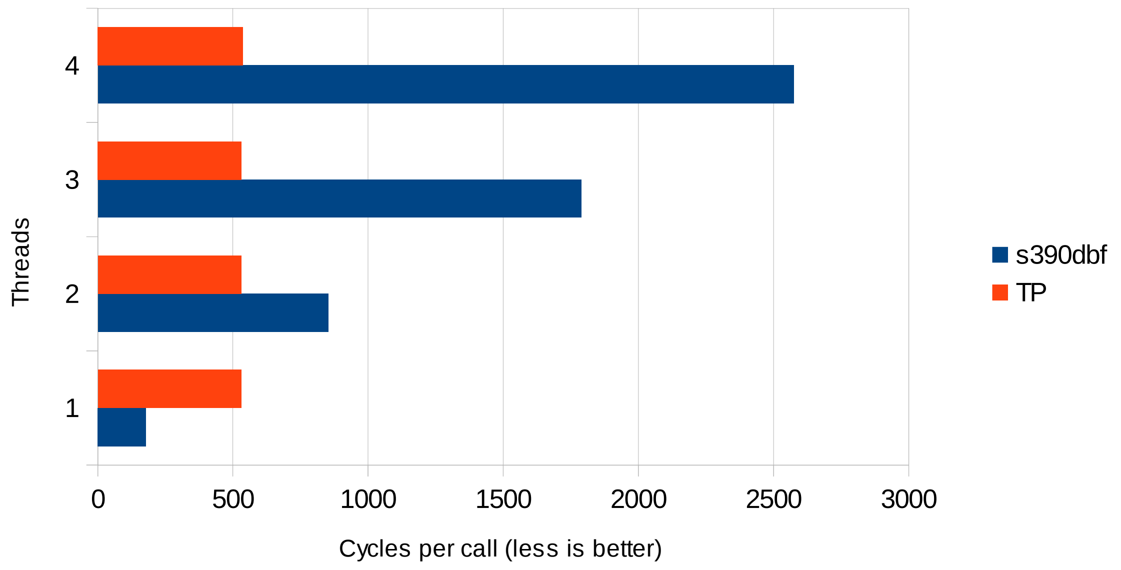
Test case:
Trace "unsigned long" value
100.000.000 times

Comparison - Performance

- **Test case: Trace “unsigned long” value (100.000.000 loops)**
- **Used perf tool with s390 hardware counters**
- **Test System:**
  - LPAR on EC12
  - Linux 3.14
  - gcc 4.3.4
- **CPUs on same Chip**
- **Disabled trace:**
  - s390dbf: “if (level)” -> 2 cycles
  - TP: jumplabel NOP -> 1 cycle

Comparison - Performance

	Threads	Instr	Cycles
TP	4	596	537
DBF	4	127	2575
TP	3	596	532
DBF	3	123	1789
TP	2	595	531
DBF	2	118	853
TP	1	595	531
DBF	1	116	179



- **s390dbf single threaded: 3 x faster**
- **s390dbf spinlock: Cachline pingpong consumes cycles**
- **How likely is lock contention?**

Tracepoints - ABI considerations

- Not 100% clear
- If there is userspace that exploits tracepoints, it is considered as ABI
- If there is no “known” userspace, tracepoints can be changed and removed
- With *libtracevent* it should always be possible to add new fields to events

Comparison

Category	s390dbf	Tracepoints
One common trace mechanism	no	yes
Trace streaming	no	trace_pipe
Per-CPU buffers	no	yes
Memory consumption buffers	global buffer	per-CPU buffers
Memory consumption static	no	TP structures / code
Single copy of structures, not intermediate buffer necessary	no	works
Single thread costs	179 cycles	531 cycles
Multi thread costs	853/1789/2575 cycles	531 cycles
Disabled tracepoint costs	2 cycles (plus memory acc)	1 cycle
Dump support	yes	main buffer
Combine other tracers with tracepoints	needs tooling	yes
Combine different trace subsystems	needs tooling	yes
Tooling	crash	crash, trace-cmd, perf
Lines of code per trace point	1	several
TP are ABI?	no	yes/sometimes

Tracepoints - Requirements

- **API for boot time:**
  - Allow to create instances (equivalent to debug areas)
  - Allow to enable tracepoints in instances
  - Allow to set event filters for instances (to separate s390dbfs)
    - E.g. for each new device a new set of tracepoints
    - Fast enough?
- **Memory usage:**
  - Single buffer option for slow traces?
  - Single backend buffer behind per-CPU buffers?
- **crash trace plugin: trace.so**
  - Allow to access instances
- **Increase single thread performance?**

Thank you!