Leveling Up
My Linux Kernel Contributions

- Troubleshooting the kernel panic

Juhee Kang

Speaker

Juhee Kang (claudiajkang@gmail.com)

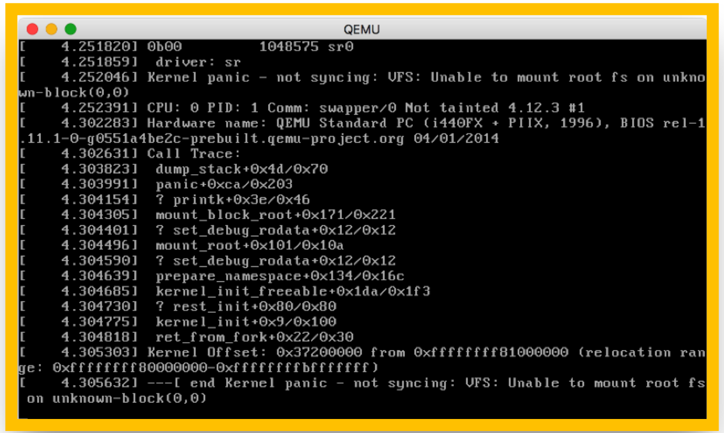
Open-Source Developer

Project: Linux Kernel Networking Stack, Kubernetes i18n

Interest: Kernel, Cloud, Network, Backend, etc...

Special Lecture at Universities
- Open-Source Software Development Mechanism.

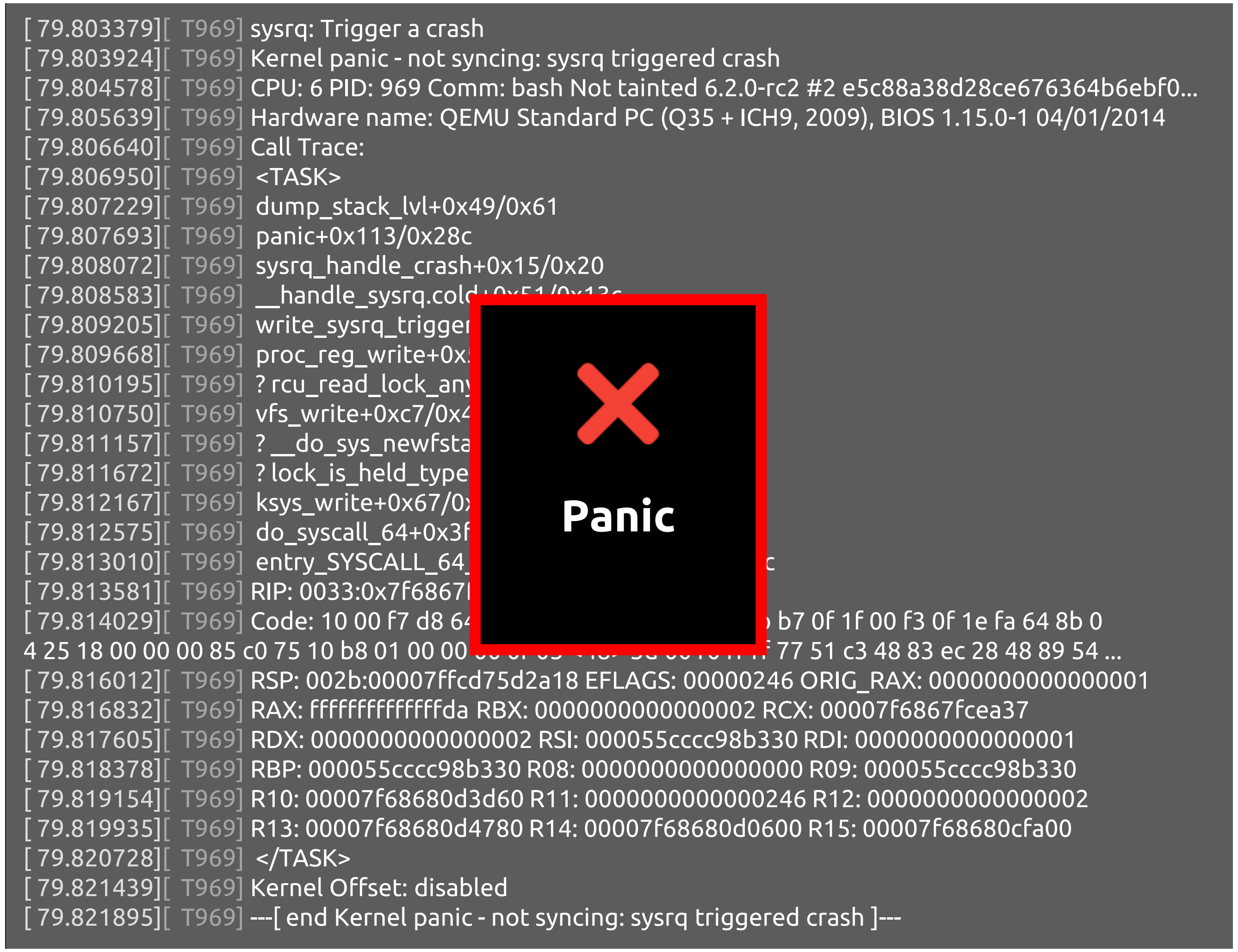
Sometimes kernel panic occurs



```
[    4.251820] 0b00          1048575 sr0
[    4.251859]  driver: sr
[    4.252046] Kernel panic - not syncing: VFS: Unable to mount root fs on unkno
wn-block(0,0)
[    4.252391] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.12.3 #1
[    4.302283] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS rel-1
.11.1-0-g0551a4be2c-prebuilt.qemu-project.org 04/01/2014
[    4.302631] Call Trace:
[    4.303823]  dump_stack+0x4d/0x70
[    4.303991]  panic+0xca/0x203
[    4.304154]  ? printk+0x3e/0x46
[    4.304305]  mount_block_root+0x171/0x221
[    4.304401]  ? set_debug_rodata+0x12/0x12
[    4.304496]  mount_root+0x101/0x10a
[    4.304590]  ? set_debug_rodata+0x12/0x12
[    4.304639]  prepare_namespace+0x134/0x16c
[    4.304685]  kernel_init_freeable+0x1da/0x1f3
[    4.304730]  ? rest_init+0x80/0x80
[    4.304775]  kernel_init+0x9/0x100
[    4.304818]  ret_from_fork+0x22/0x30
[    4.305303] Kernel Offset: 0x37200000 from 0xffffffff81000000 (relocation ran
ge: 0xffffffff80000000-0xffffffffbfffffff)
[    4.305632] ---[ end Kernel panic - not syncing: VFS: Unable to mount root fs
 on unknown-block(0,0)
```

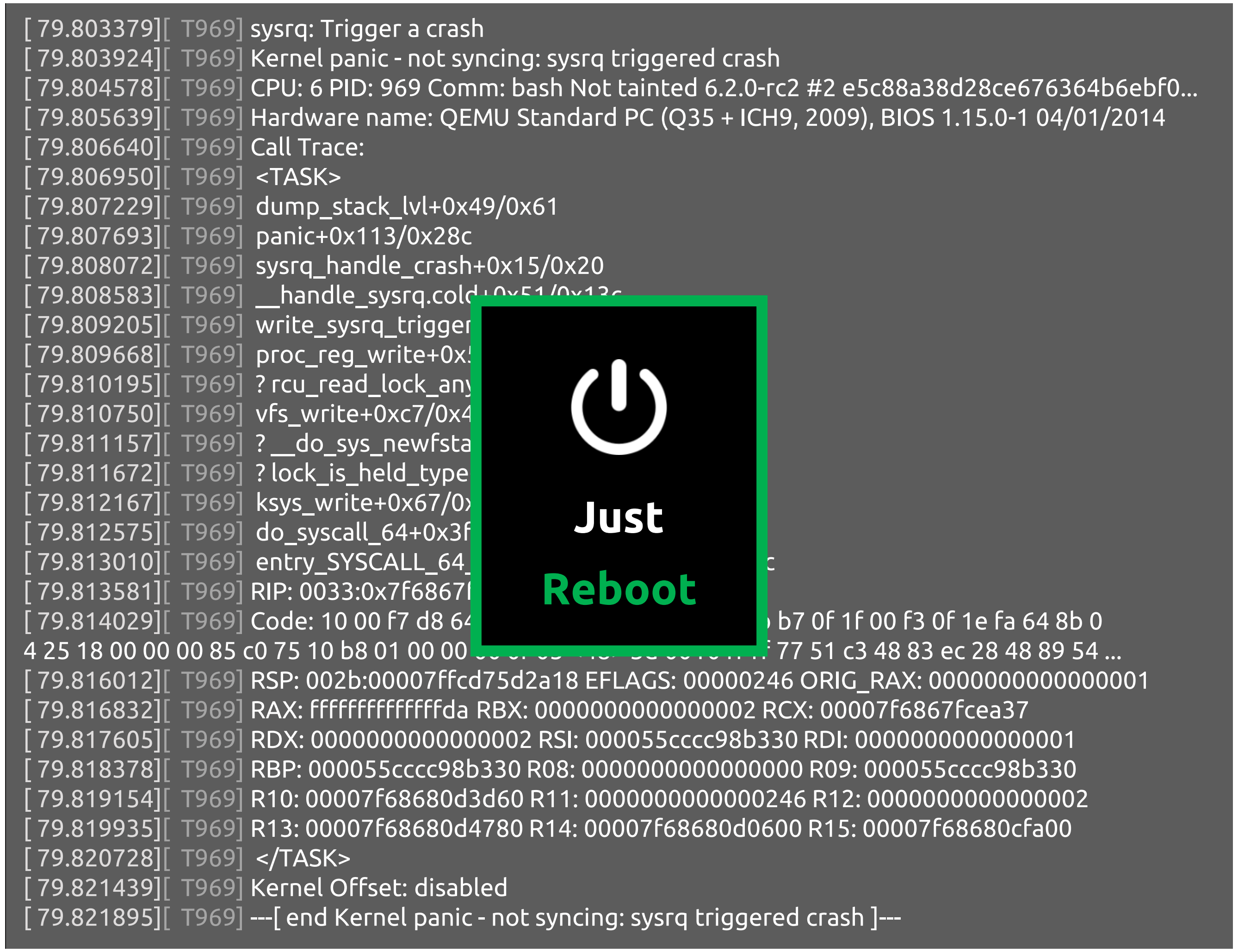
```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger+0x42/0x50
[ 79.809668][  T969]  proc_reg_write+0x54/0xa0
[ 79.810195][  T969]  ? rcu_read_lock_any_held+0x79/0xa0
[ 79.810750][  T969]  vfs_write+0xc7/0x4a0
[ 79.811157][  T969]  ? __do_sys_newfstatat+0x35/0x60
[ 79.811672][  T969]  ? lock_is_held_type+0xe1/0x140
[ 79.812167][  T969]  ksys_write+0x67/0xf0
[ 79.812575][  T969]  do_syscall_64+0x3f/0x90
[ 79.813010][  T969]  entry_SYSCALL_64_after_hwframe+0x72/0xdc
[ 79.813581][  T969] RIP: 0033:0x7f6867fcea37
[ 79.814029][  T969] Code: 10 00 f7 d8 64 89 02 48 c7 c0 ff ff ff ff eb b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Normally I solved the kernel panic



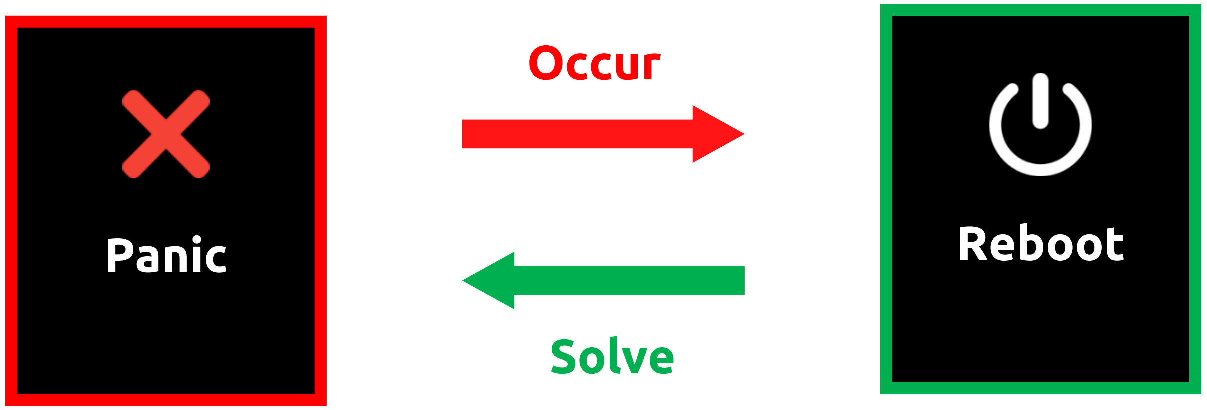
```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger
[ 79.809668][  T969]  proc_reg_write+0x
[ 79.810195][  T969]  ? rcu_read_lock_any
[ 79.810750][  T969]  vfs_write+0xc7/0x4
[ 79.811157][  T969]  ? __do_sys_newfsta
[ 79.811672][  T969]  ? lock_is_held_type
[ 79.812167][  T969]  ksys_write+0x67/0x
[ 79.812575][  T969]  do_syscall_64+0x3f
[ 79.813010][  T969]  entry_SYSCALL_64_                    c
[ 79.813581][  T969] RIP: 0033:0x7f6867
[ 79.814029][  T969] Code: 10 00 f7 d8 64                  b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00                 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Normally I solved the kernel panic



```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold
[ 79.809205][  T969]  write_sysrq_trigger
[ 79.809668][  T969]  proc_reg_write+0x
[ 79.810195][  T969]  ? rcu_read_lock_any
[ 79.810750][  T969]  vfs_write+0xc7/0x4
[ 79.811157][  T969]  ? __do_sys_newfsta
[ 79.811672][  T969]  ? lock_is_held_type
[ 79.812167][  T969]  ksys_write+0x67/0x
[ 79.812575][  T969]  do_syscall_64+0x3f
[ 79.813010][  T969]  entry_SYSCALL_64_                       c
[ 79.813581][  T969] RIP: 0033:0x7f6867f
[ 79.814029][  T969] Code: 10 00 f7 d8 64                     b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00                    77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Normally I solved the kernel panic



Infinite loop…

Troubleshooting kernel panic?

Is there any effective way of troubleshooting kernel panic?

Session's goal

Step through kernel panic debugging with example!

How to cause kernel panic with a single command?

Easiest way to raise kernel panic

```
# echo c > /proc/sysrq-trigger
```

https:///www.kernel.org/doc/html/v6.2-rc2/admin-guide/sysrq.html

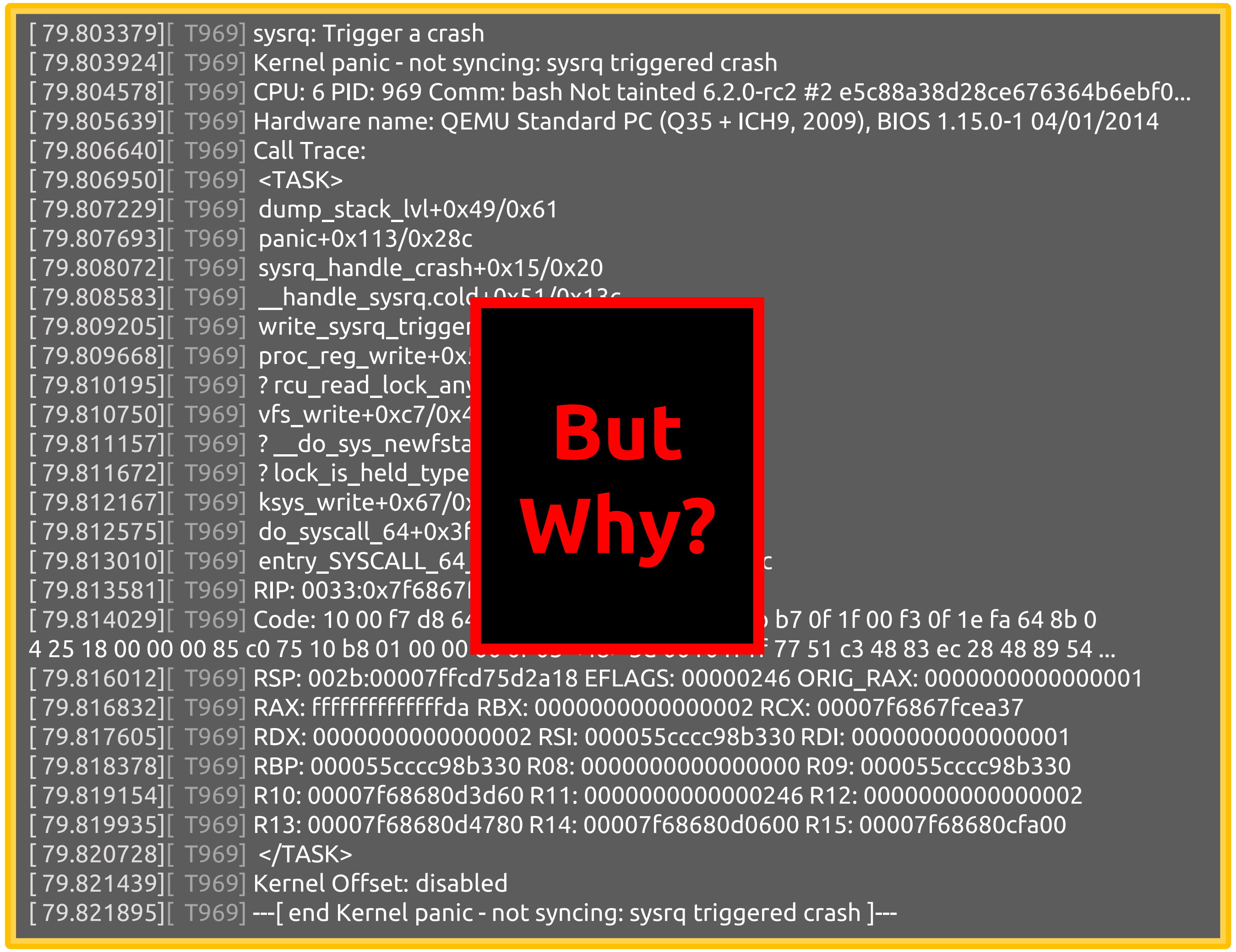
Raise Kernel panic

```
# echo c > /proc/sysrq-trigger
```

```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger+0x42/0x50
[ 79.809668][  T969]  proc_reg_write+0x54/0xa0
[ 79.810195][  T969]  ? rcu_read_lock_any_held+0x79/0xa0
[ 79.810750][  T969]  vfs_write+0xc7/0x4a0
[ 79.811157][  T969]  ? __do_sys_newfstatat+0x35/0x60
[ 79.811672][  T969]  ? lock_is_held_type+0xe1/0x140
[ 79.812167][  T969]  ksys_write+0x67/0xf0
[ 79.812575][  T969]  do_syscall_64+0x3f/0x90
[ 79.813010][  T969]  entry_SYSCALL_64_after_hwframe+0x72/0xdc
[ 79.813581][  T969] RIP: 0033:0x7f6867fcea37
[ 79.814029][  T969] Code: 10 00 f7 d8 64 89 02 48 c7 c0 ff ff ff ff eb b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Raise Kernel panic

```
# echo c > /proc/sysrq-trigger
```



```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold
[ 79.809205][  T969]  write_sysrq_trigger
[ 79.809668][  T969]  proc_reg_write+0x
[ 79.810195][  T969]  ? rcu_read_lock_any
[ 79.810750][  T969]  vfs_write+0xc7/0x4
[ 79.811157][  T969]  ? __do_sys_newfsta
[ 79.811672][  T969]  ? lock_is_held_type
[ 79.812167][  T969]  ksys_write+0x67/0x
[ 79.812575][  T969]  do_syscall_64+0x3f
[ 79.813010][  T969]  entry_SYSCALL_64_                 c
[ 79.813581][  T969] RIP: 0033:0x7f6867f
[ 79.814029][  T969] Code: 10 00 f7 d8 64                  b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00                  77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

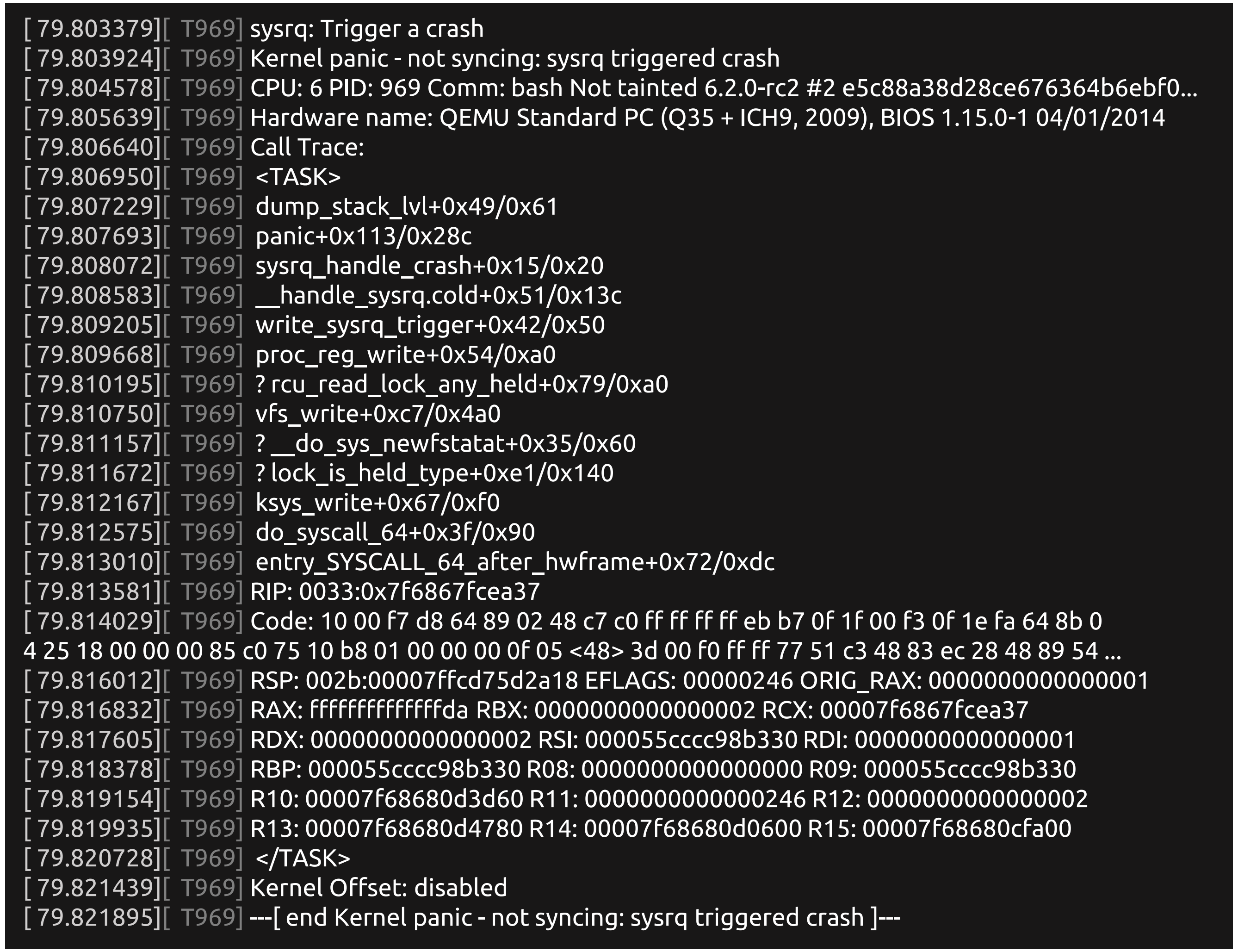
But Why?

Analysis kernel panic log

First, let's look at the Kernel Panic log

Analysis kernel panic log

① Skim through Kernel Panic log



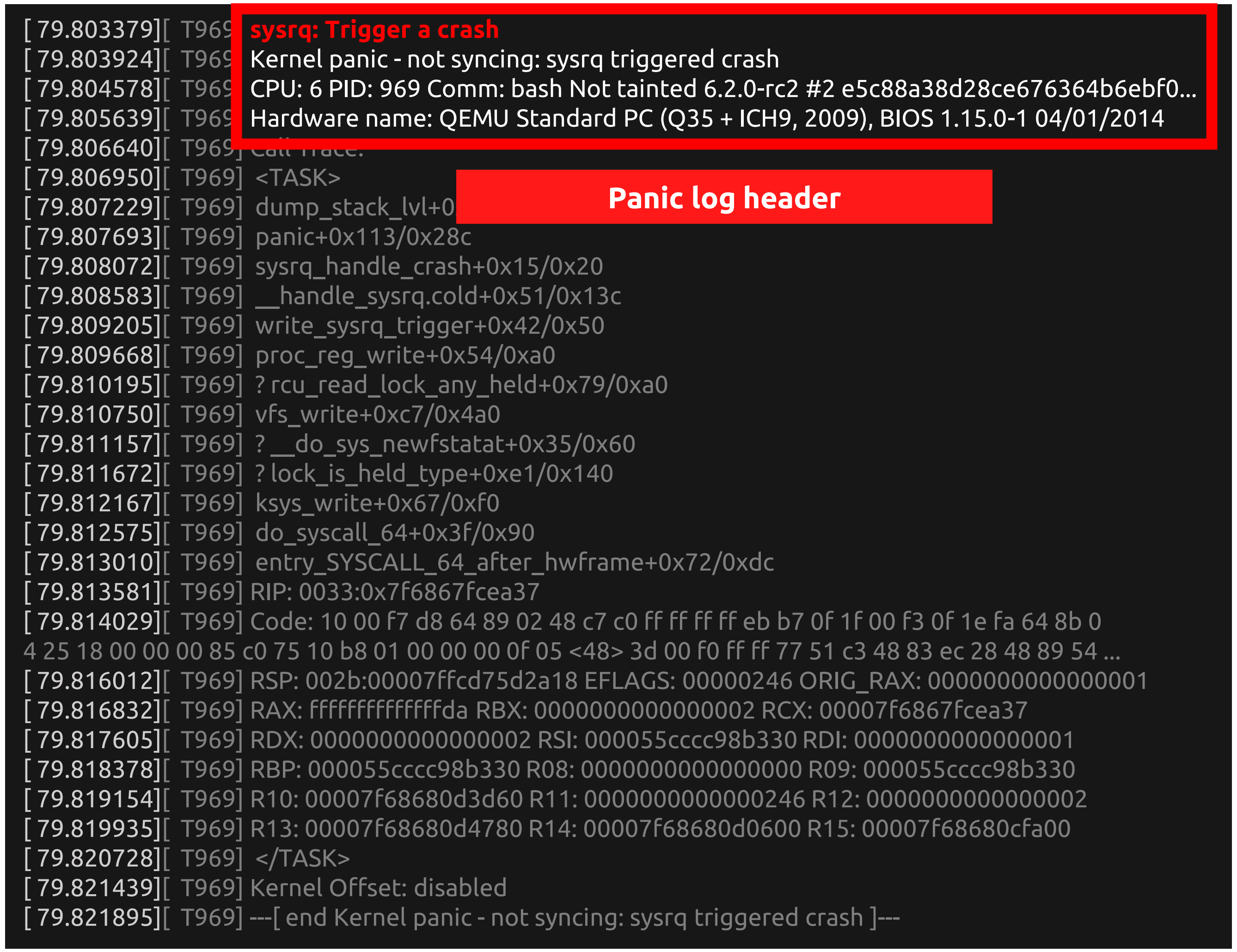
```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger+0x42/0x50
[ 79.809668][  T969]  proc_reg_write+0x54/0xa0
[ 79.810195][  T969]  ? rcu_read_lock_any_held+0x79/0xa0
[ 79.810750][  T969]  vfs_write+0xc7/0x4a0
[ 79.811157][  T969]  ? __do_sys_newfstatat+0x35/0x60
[ 79.811672][  T969]  ? lock_is_held_type+0xe1/0x140
[ 79.812167][  T969]  ksys_write+0x67/0xf0
[ 79.812575][  T969]  do_syscall_64+0x3f/0x90
[ 79.813010][  T969]  entry_SYSCALL_64_after_hwframe+0x72/0xdc
[ 79.813581][  T969] RIP: 0033:0x7f6867fcea37
[ 79.814029][  T969] Code: 10 00 f7 d8 64 89 02 48 c7 c0 ff ff ff ff eb b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Analysis kernel panic log

① **Skim through Kernel Panic log**

Panic log header provides the abstraction about the crash

ex) Cause of the panic



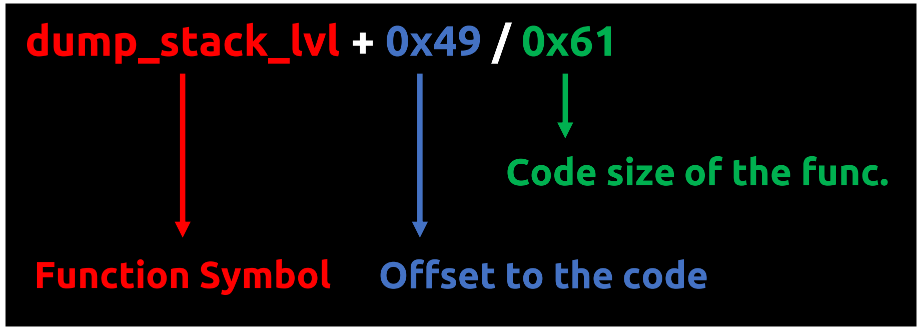
```
[ 79.803379][ T969 sysrq: Trigger a crash
[ 79.803924][ T969 Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][ T969 CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][ T969 Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][ T969] Call Trace:
[ 79.806950][ T969]  <TASK>
[ 79.807229][ T969]  dump_stack_lvl+0
[ 79.807693][ T969]  panic+0x113/0x28c
[ 79.808072][ T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][ T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][ T969]  write_sysrq_trigger+0x42/0x50
[ 79.809668][ T969]  proc_reg_write+0x54/0xa0
[ 79.810195][ T969]  ? rcu_read_lock_any_held+0x79/0xa0
[ 79.810750][ T969]  vfs_write+0xc7/0x4a0
[ 79.811157][ T969]  ? __do_sys_newfstatat+0x35/0x60
[ 79.811672][ T969]  ? lock_is_held_type+0xe1/0x140
[ 79.812167][ T969]  ksys_write+0x67/0xf0
[ 79.812575][ T969]  do_syscall_64+0x3f/0x90
[ 79.813010][ T969]  entry_SYSCALL_64_after_hwframe+0x72/0xdc
[ 79.813581][ T969] RIP: 0033:0x7f6867fcea37
[ 79.814029][ T969] Code: 10 00 f7 d8 64 89 02 48 c7 c0 ff ff ff ff eb b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][ T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][ T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][ T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][ T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][ T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][ T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][ T969]  </TASK>
[ 79.821439][ T969] Kernel Offset: disabled
[ 79.821895][ T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

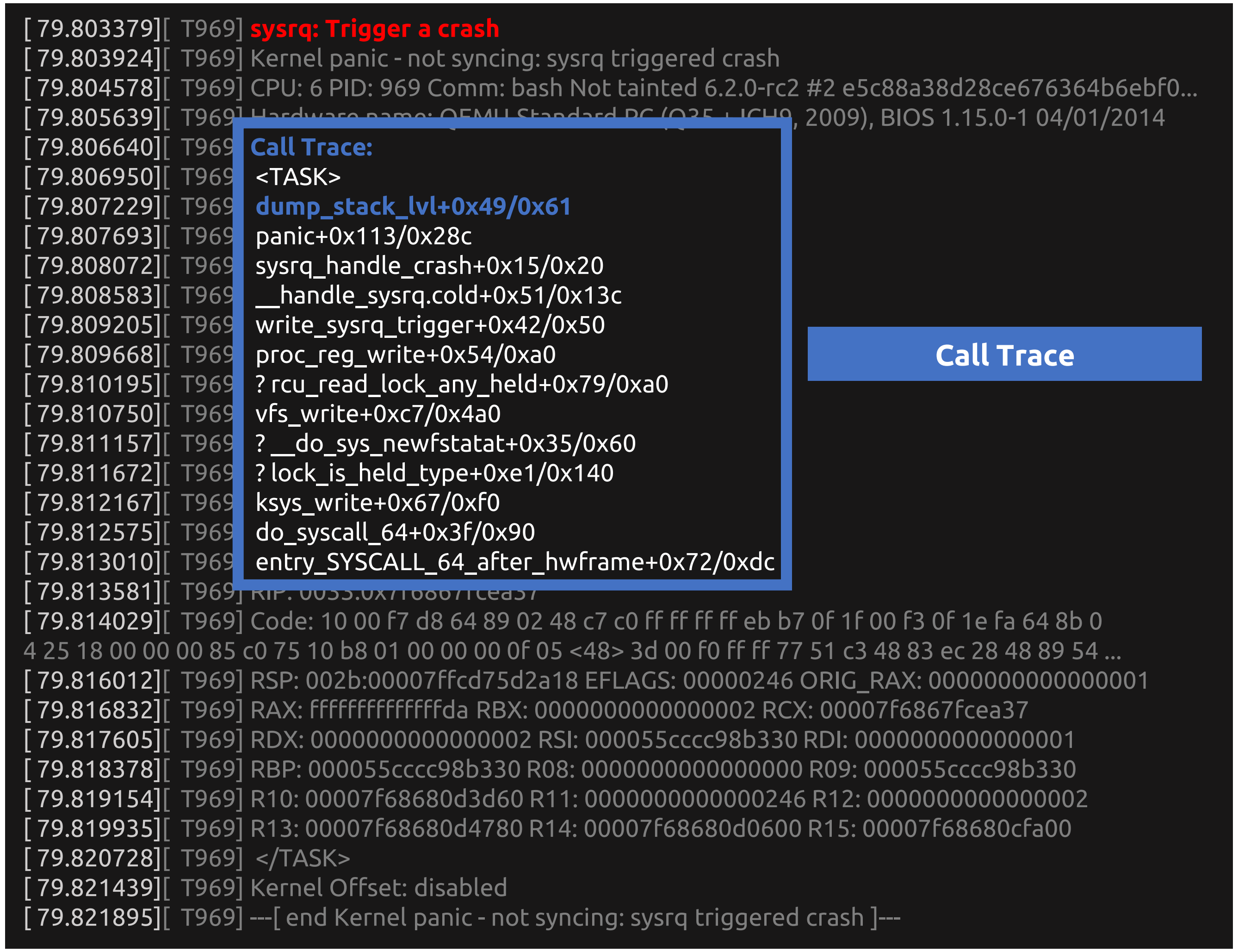
Analysis kernel panic log

① Skim through Kernel Panic log

Panic log header provides the abstraction about the crash

Call Trace provides the context information about the crash




```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger+0x42/0x50
[ 79.809668][  T969]  proc_reg_write+0x54/0xa0
[ 79.810195][  T969]  ? rcu_read_lock_any_held+0x79/0xa0
[ 79.810750][  T969]  vfs_write+0xc7/0x4a0
[ 79.811157][  T969]  ? __do_sys_newfstatat+0x35/0x60
[ 79.811672][  T969]  ? lock_is_held_type+0xe1/0x140
[ 79.812167][  T969]  ksys_write+0x67/0xf0
[ 79.812575][  T969]  do_syscall_64+0x3f/0x90
[ 79.813010][  T969]  entry_SYSCALL_64_after_hwframe+0x72/0xdc
[ 79.813581][  T969] RIP: 0033:0x7f6867fcea37
[ 79.814029][  T969] Code: 10 00 f7 d8 64 89 02 48 c7 c0 ff ff ff ff eb b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

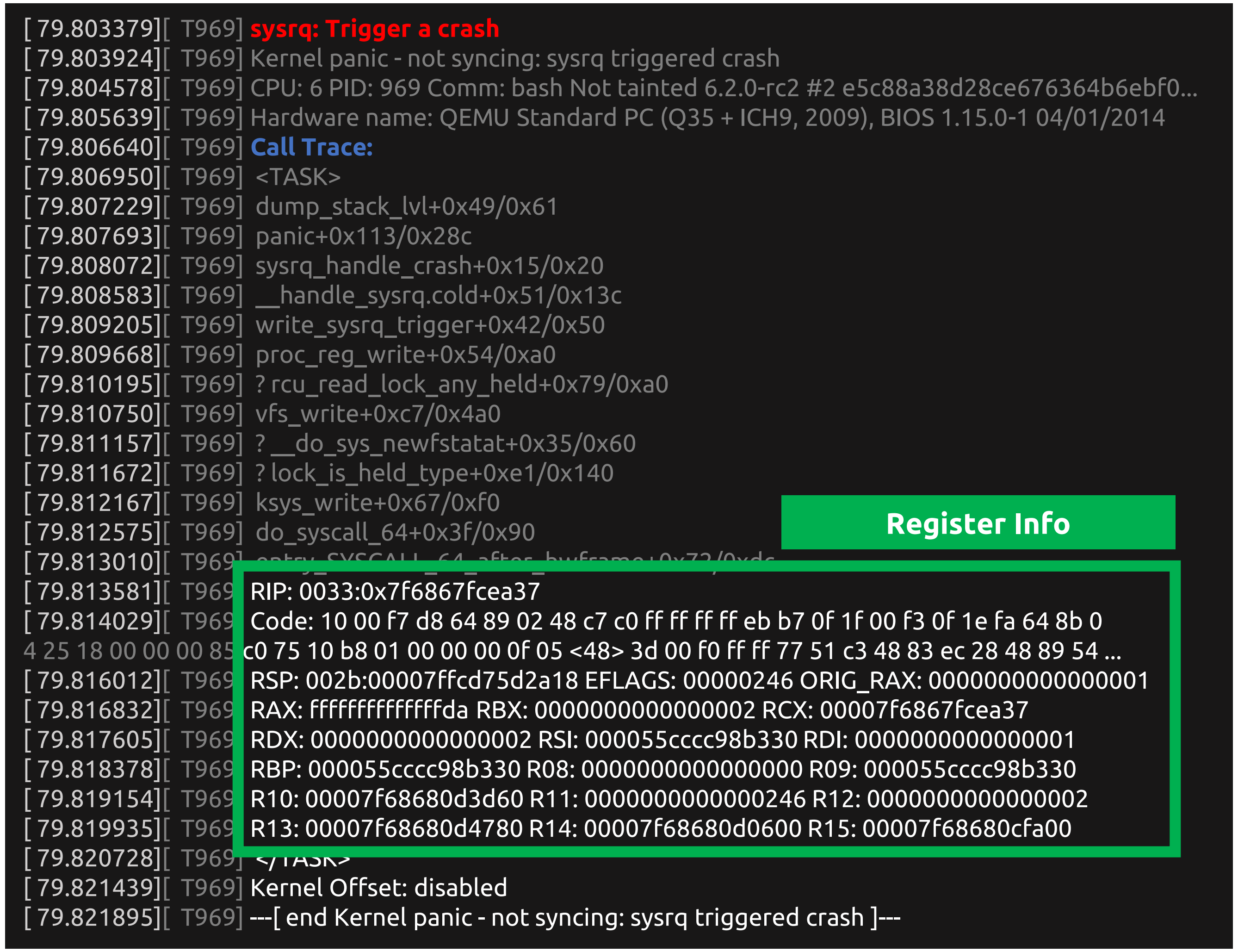
Analysis kernel panic log

① Skim through Kernel Panic log

Panic log header provides the abstraction about the crash

Call Trace provides the context information about the crash

Register Information provides the current executing dump of CPU registers



```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger+0x42/0x50
[ 79.809668][  T969]  proc_reg_write+0x54/0xa0
[ 79.810195][  T969]  ? rcu_read_lock_any_held+0x79/0xa0
[ 79.810750][  T969]  vfs_write+0xc7/0x4a0
[ 79.811157][  T969]  ? __do_sys_newfstatat+0x35/0x60
[ 79.811672][  T969]  ? lock_is_held_type+0xe1/0x140
[ 79.812167][  T969]  ksys_write+0x67/0xf0
[ 79.812575][  T969]  do_syscall_64+0x3f/0x90
[ 79.813010][  T969]  entry_SYSCALL_64_after_hwframe+0x72/0xdc
[ 79.813581][  T969  RIP: 0033:0x7f6867fcea37
[ 79.814029][  T969  Code: 10 00 f7 d8 64 89 02 48 c7 c0 ff ff ff ff eb b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969  RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969  RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969  RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969  RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969  R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969  R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Analysis kernel panic log

① Skim through Kernel Panic log

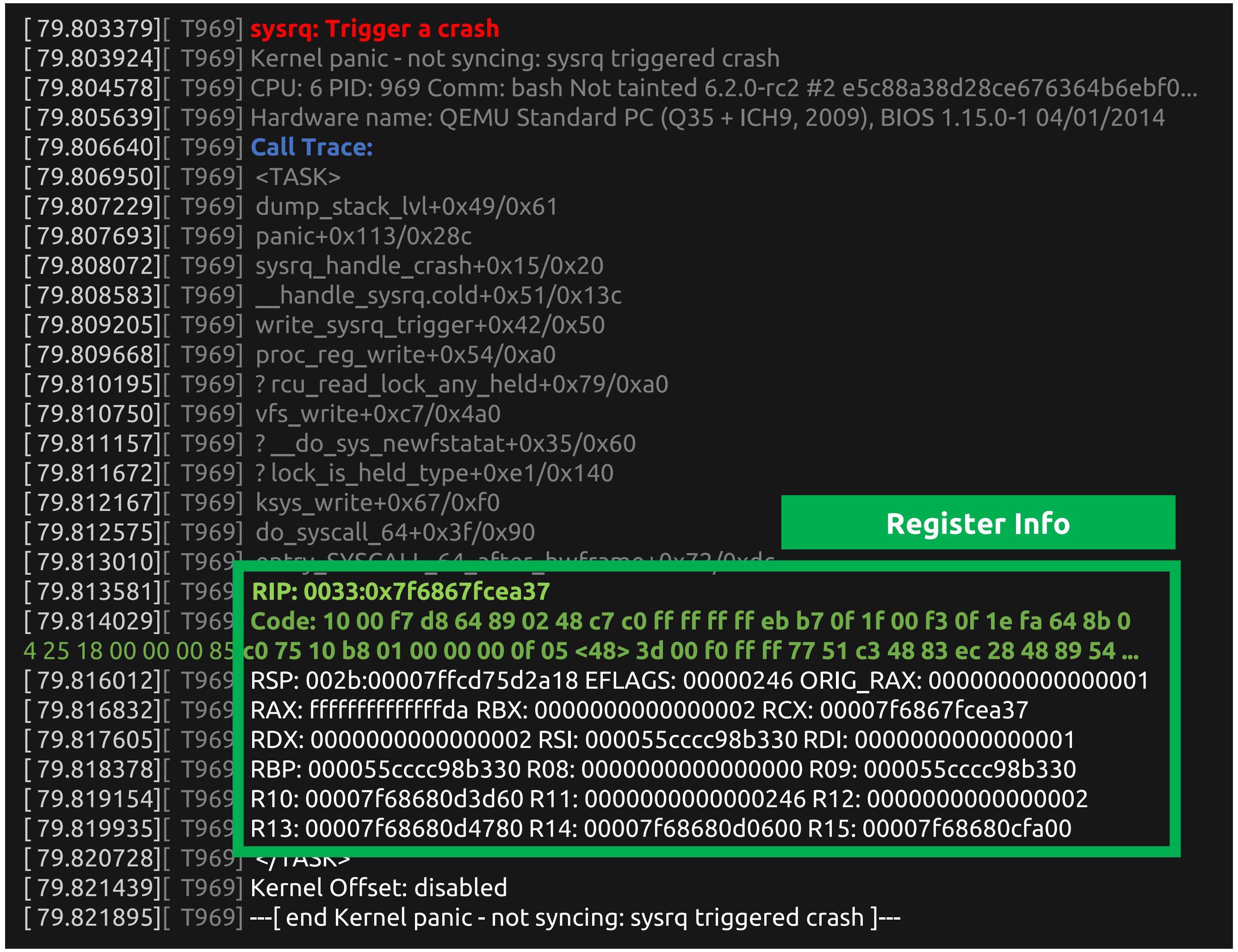
Panic log header provides the abstraction about the crash

Call Trace provides the context information about the crash

Register Information provides the current executing dump of CPU registers

RIP holds the current executing instruction

Code includes the current executing code information

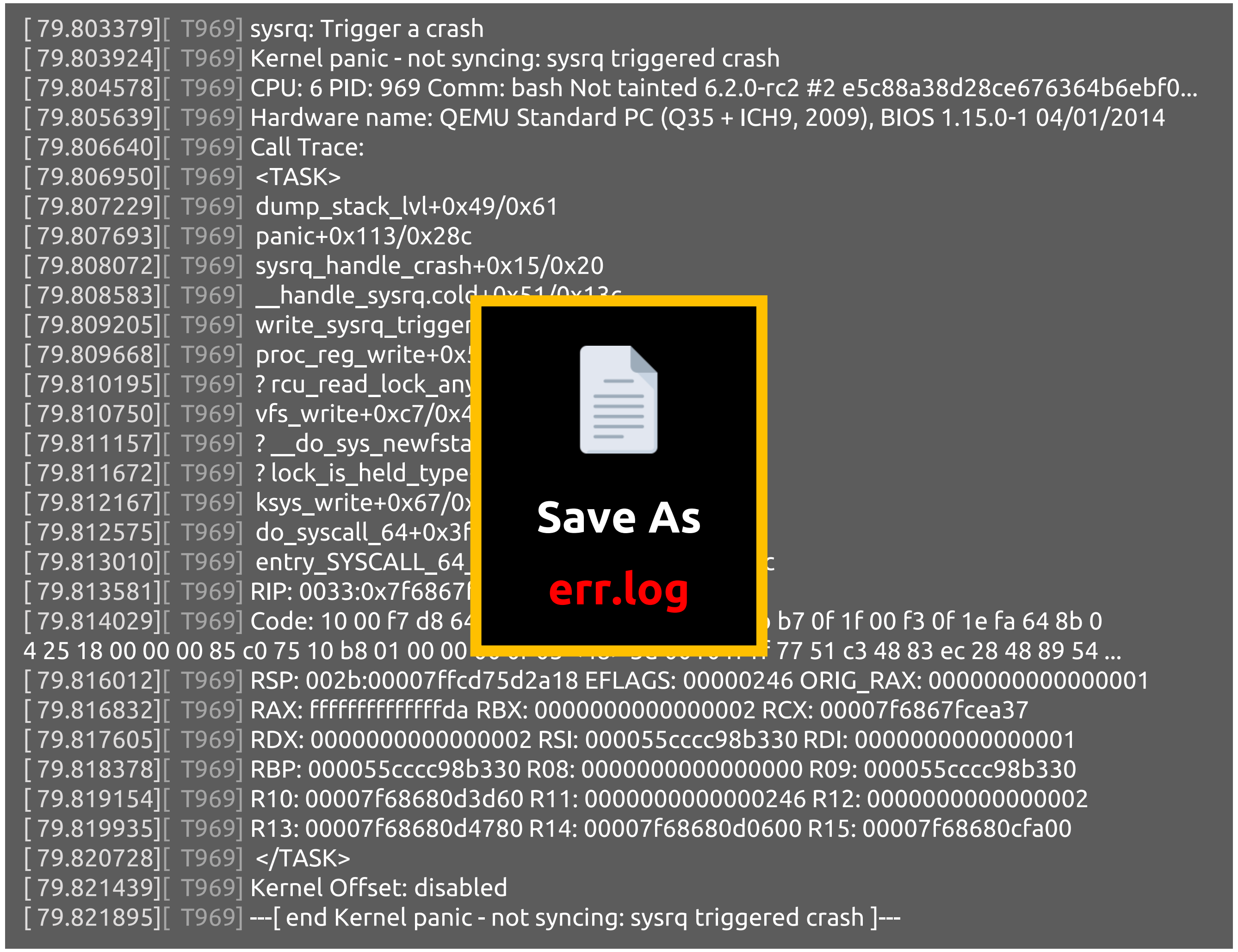


```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger+0x42/0x50
[ 79.809668][  T969]  proc_reg_write+0x54/0xa0
[ 79.810195][  T969]  ? rcu_read_lock_any_held+0x79/0xa0
[ 79.810750][  T969]  vfs_write+0xc7/0x4a0
[ 79.811157][  T969]  ? __do_sys_newfstatat+0x35/0x60
[ 79.811672][  T969]  ? lock_is_held_type+0xe1/0x140
[ 79.812167][  T969]  ksys_write+0x67/0xf0
[ 79.812575][  T969]  do_syscall_64+0x3f/0x90
[ 79.813010][  T969]  entry_SYSCALL_64_after_hwframe+0x72/0xdc
[ 79.813581][  T969 RIP: 0033:0x7f6867fcea37
[ 79.814029][  T969 Code: 10 00 f7 d8 64 89 02 48 c7 c0 ff ff ff ff eb b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969 RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969 RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969 RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969 RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969 R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969 R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Analysis kernel panic log

① Skim through Kernel Panic log

For call trace analysis, save the panic log

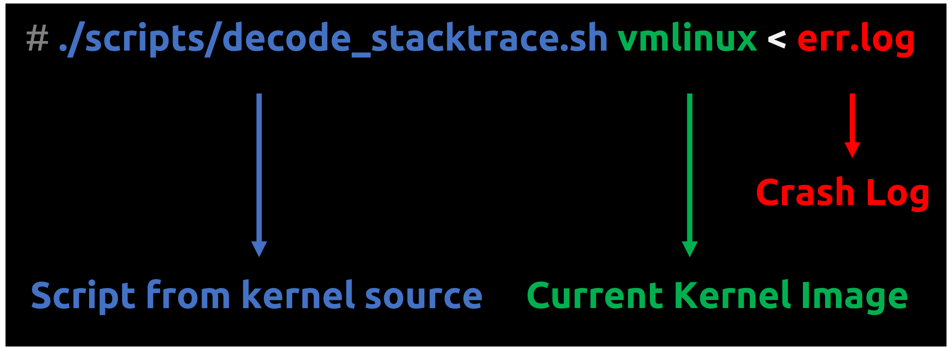


```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger
[ 79.809668][  T969]  proc_reg_write+0x
[ 79.810195][  T969]  ? rcu_read_lock_any
[ 79.810750][  T969]  vfs_write+0xc7/0x4
[ 79.811157][  T969]  ? __do_sys_newfsta
[ 79.811672][  T969]  ? lock_is_held_type
[ 79.812167][  T969]  ksys_write+0x67/0x
[ 79.812575][  T969]  do_syscall_64+0x3f
[ 79.813010][  T969]  entry_SYSCALL_64_                              c
[ 79.813581][  T969] RIP: 0033:0x7f6867f
[ 79.814029][  T969] Code: 10 00 f7 d8 64                             b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00                            77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

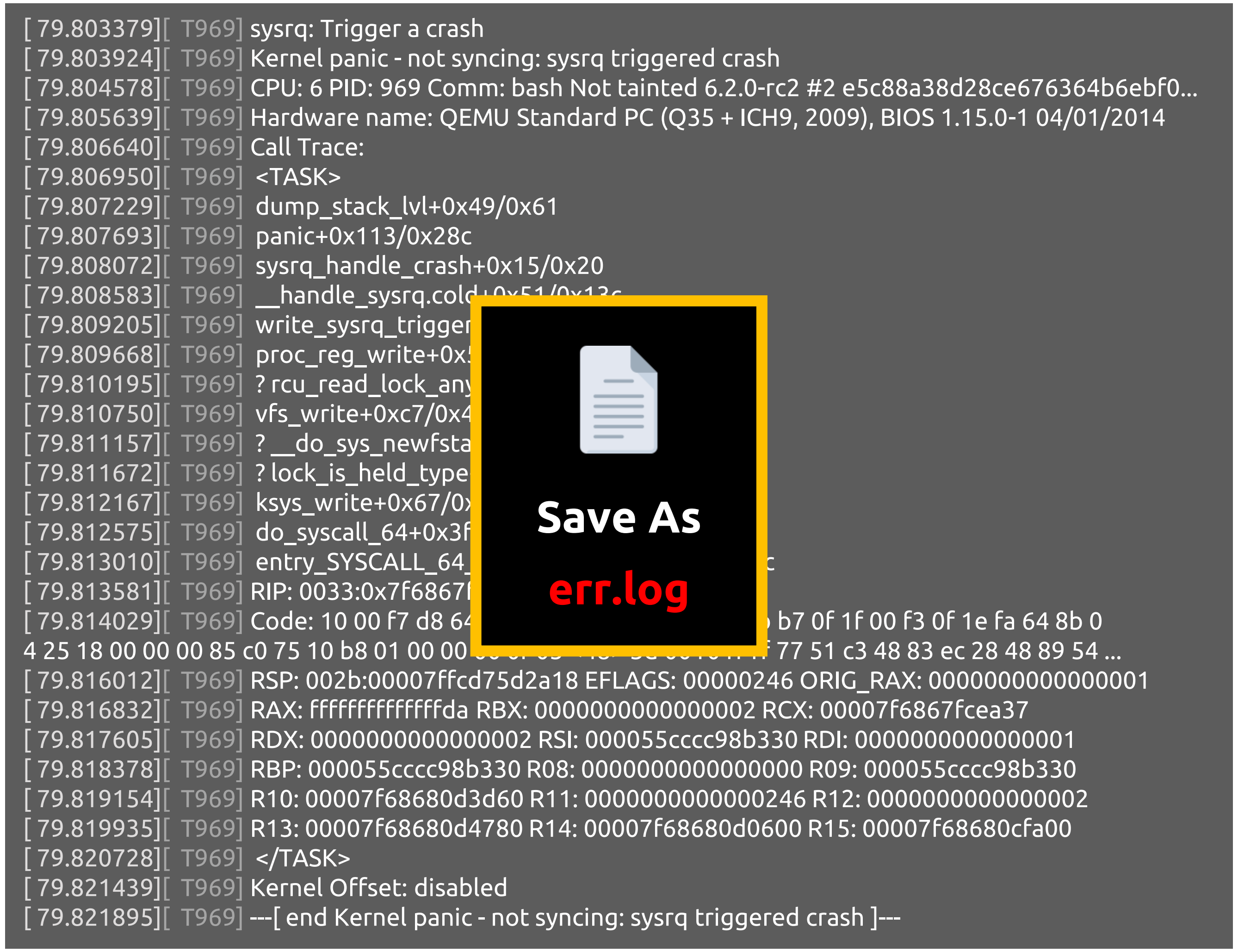
Analysis kernel panic log

① Skim through Kernel Panic log

② **Decode call trace**



```
# ./scripts/decode_stacktrace.sh vmlinux < err.log
```

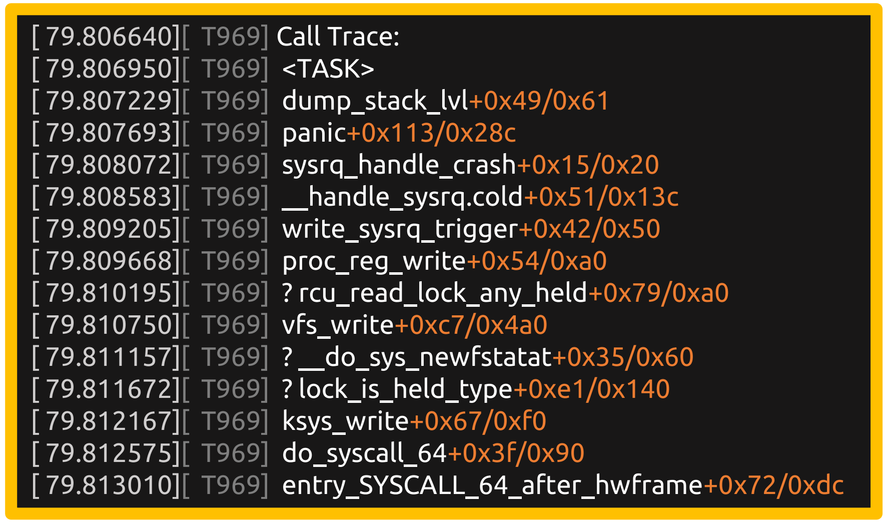


```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger
[ 79.809668][  T969]  proc_reg_write+0x5
[ 79.810195][  T969]  ? rcu_read_lock_any
[ 79.810750][  T969]  vfs_write+0xc7/0x4
[ 79.811157][  T969]  ? __do_sys_newfsta
[ 79.811672][  T969]  ? lock_is_held_type
[ 79.812167][  T969]  ksys_write+0x67/0x
[ 79.812575][  T969]  do_syscall_64+0x3f
[ 79.813010][  T969]  entry_SYSCALL_64_                              c
[ 79.813581][  T969] RIP: 0033:0x7f6867f
[ 79.814029][  T969] Code: 10 00 f7 d8 64                          b7 0f 1f 00 f3 0f 1e fa 64 8b 0
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00                          f 77 51 c3 48 83 ec 28 48 89 54 ...
[ 79.816012][  T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG_RAX: 0000000000000001
[ 79.816832][  T969] RAX: ffffffffffffffda RBX: 0000000000000002 RCX: 00007f6867fcea37
[ 79.817605][  T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][  T969] RBP: 000055cccc98b330 R08: 0000000000000000 R09: 000055cccc98b330
[ 79.819154][  T969] R10: 00007f68680d3d60 R11: 0000000000000246 R12: 0000000000000002
[ 79.819935][  T969] R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

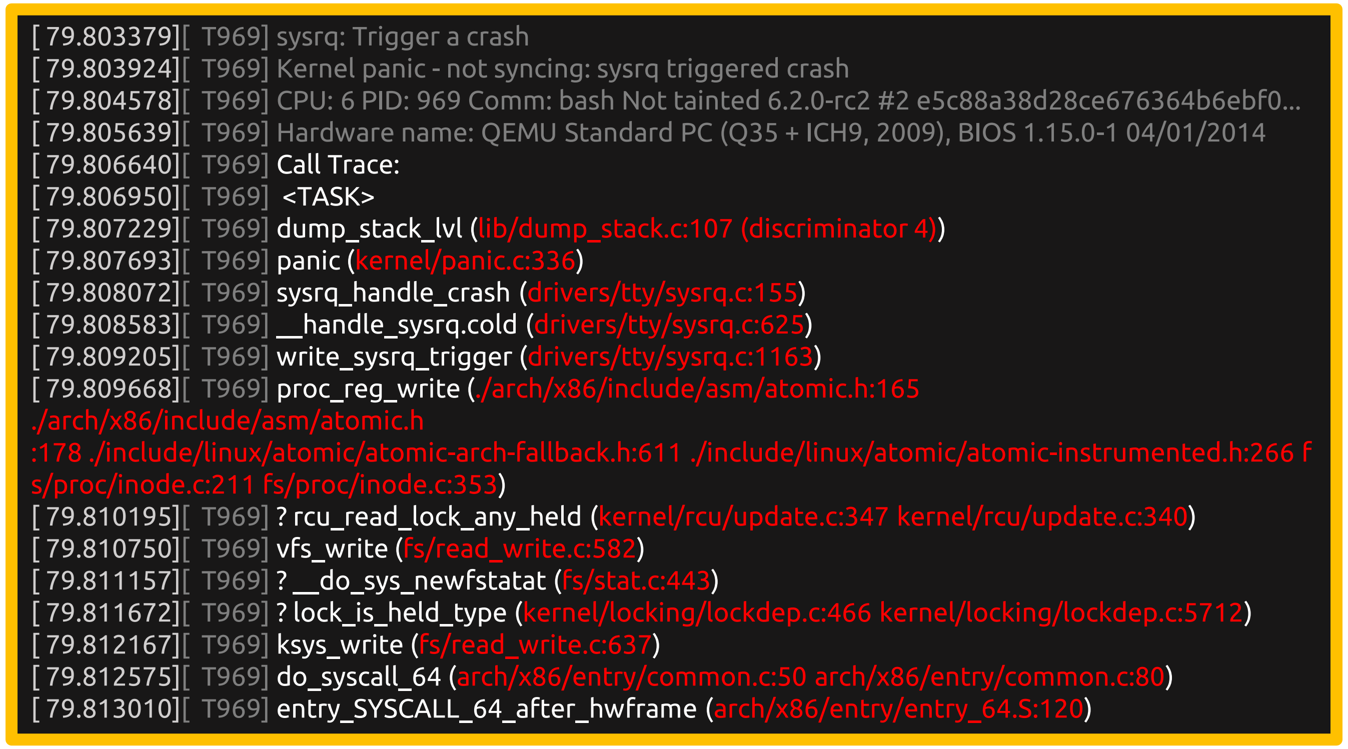
Analysis kernel panic log

① Skim through Kernel Panic log

② Decode call trace



```
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969]  dump_stack_lvl+0x49/0x61
[ 79.807693][  T969]  panic+0x113/0x28c
[ 79.808072][  T969]  sysrq_handle_crash+0x15/0x20
[ 79.808583][  T969]  __handle_sysrq.cold+0x51/0x13c
[ 79.809205][  T969]  write_sysrq_trigger+0x42/0x50
[ 79.809668][  T969]  proc_reg_write+0x54/0xa0
[ 79.810195][  T969]  ? rcu_read_lock_any_held+0x79/0xa0
[ 79.810750][  T969]  vfs_write+0xc7/0x4a0
[ 79.811157][  T969]  ? __do_sys_newfstatat+0x35/0x60
[ 79.811672][  T969]  ? lock_is_held_type+0xe1/0x140
[ 79.812167][  T969]  ksys_write+0x67/0xf0
[ 79.812575][  T969]  do_syscall_64+0x3f/0x90
[ 79.813010][  T969]  entry_SYSCALL_64_after_hwframe+0x72/0xdc
```



```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969] dump_stack_lvl (lib/dump_stack.c:107 (discriminator 4))
[ 79.807693][  T969] panic (kernel/panic.c:336)
[ 79.808072][  T969] sysrq_handle_crash (drivers/tty/sysrq.c:155)
[ 79.808583][  T969] __handle_sysrq.cold (drivers/tty/sysrq.c:625)
[ 79.809205][  T969] write_sysrq_trigger (drivers/tty/sysrq.c:1163)
[ 79.809668][  T969] proc_reg_write (./arch/x86/include/asm/atomic.h:165
./arch/x86/include/asm/atomic.h
:178 ./include/linux/atomic/atomic-arch-fallback.h:611 ./include/linux/atomic/atomic-instrumented.h:266 f
s/proc/inode.c:211 fs/proc/inode.c:353)
[ 79.810195][  T969] ? rcu_read_lock_any_held (kernel/rcu/update.c:347 kernel/rcu/update.c:340)
[ 79.810750][  T969] vfs_write (fs/read_write.c:582)
[ 79.811157][  T969] ? __do_sys_newfstatat (fs/stat.c:443)
[ 79.811672][  T969] ? lock_is_held_type (kernel/locking/lockdep.c:466 kernel/locking/lockdep.c:5712)
[ 79.812167][  T969] ksys_write (fs/read_write.c:637)
[ 79.812575][  T969] do_syscall_64 (arch/x86/entry/common.c:50 arch/x86/entry/common.c:80)
[ 79.813010][  T969] entry_SYSCALL_64_after_hwframe (arch/x86/entry/entry_64.S:120)
```

Shows the code information of a symbol

Let's take a deep dive into

how kernel panic actually occurs!

Analysis kernel panic log

① Skim through Kernel Panic log

② Decode call trace

③ In depth analysis

- Panic trigger command

Analysis kernel panic log

① Skim through Kernel Panic log

② Decode call trace

③ In depth analysis

- Panic trigger command

```
# echo c > /proc/sysrq-trigger
```

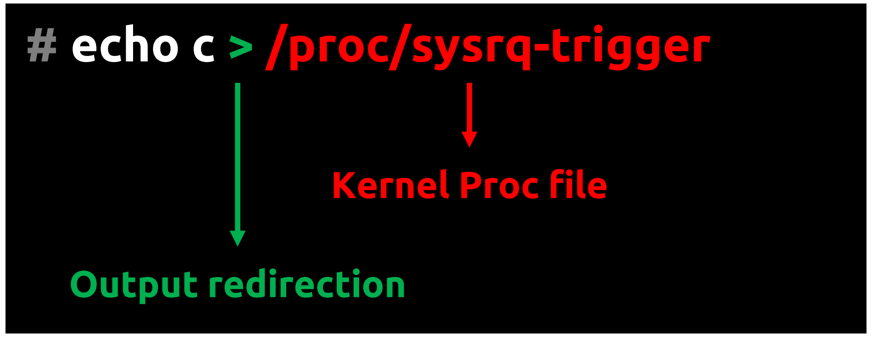
Analysis kernel panic log

① Skim through Kernel Panic log

② Decode call trace

③ **In depth analysis**

- Panic trigger command



```
# echo c > /proc/sysrq-trigger
```

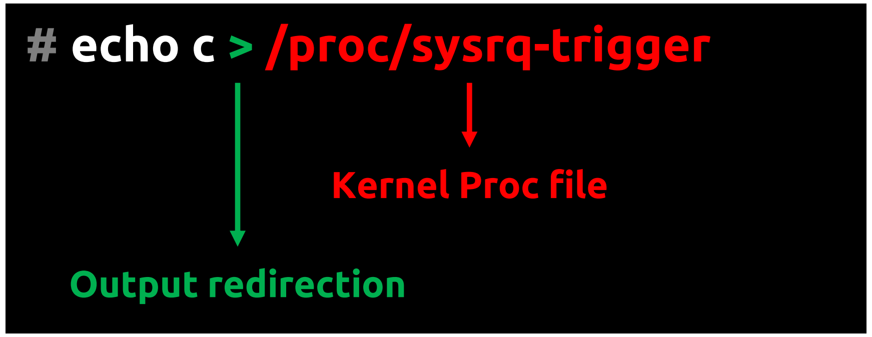
Analysis kernel panic log

① Skim through Kernel Panic log

② Decode call trace

③ In depth analysis

- Panic trigger command



```
# echo c > /proc/sysrq-trigger
```

write “c” on /proc/sysrq-trigger

Analysis kernel panic log

① Skim through Kernel Panic log

② Decode call trace

③ **In depth analysis**

- Panic trigger command

- **Analysis with Call Trace**

Analysis kernel panic log

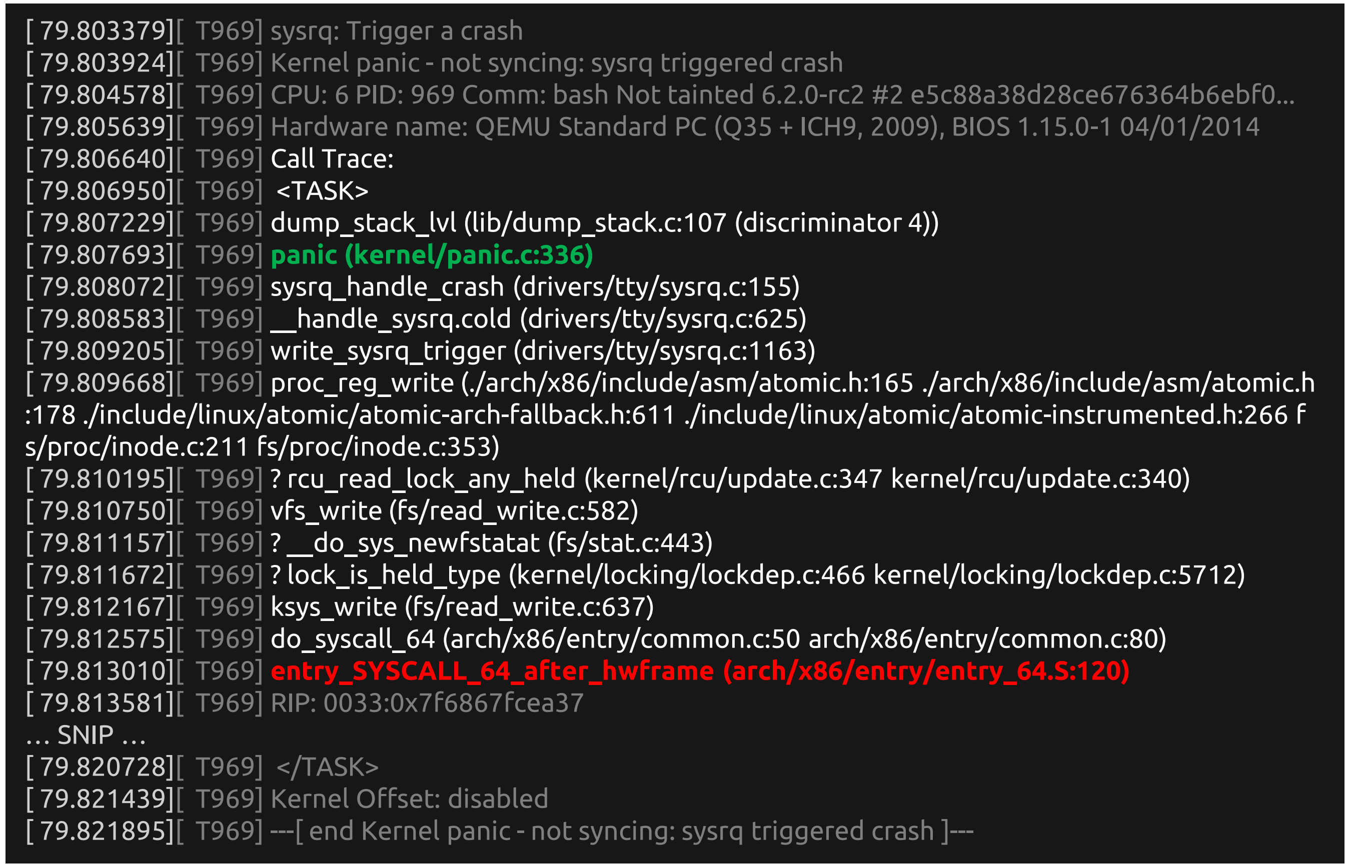
① Skim through Kernel Panic log

② Decode call trace

③ In depth analysis

- Panic trigger command

- **Analysis with Call Trace**



```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969] dump_stack_lvl (lib/dump_stack.c:107 (discriminator 4))
[ 79.807693][  T969] panic (kernel/panic.c:336)
[ 79.808072][  T969] sysrq_handle_crash (drivers/tty/sysrq.c:155)
[ 79.808583][  T969] __handle_sysrq.cold (drivers/tty/sysrq.c:625)
[ 79.809205][  T969] write_sysrq_trigger (drivers/tty/sysrq.c:1163)
[ 79.809668][  T969] proc_reg_write (./arch/x86/include/asm/atomic.h:165 ./arch/x86/include/asm/atomic.h
:178 ./include/linux/atomic/atomic-arch-fallback.h:611 ./include/linux/atomic/atomic-instrumented.h:266 f
s/proc/inode.c:211 fs/proc/inode.c:353)
[ 79.810195][  T969] ? rcu_read_lock_any_held (kernel/rcu/update.c:347 kernel/rcu/update.c:340)
[ 79.810750][  T969] vfs_write (fs/read_write.c:582)
[ 79.811157][  T969] ? __do_sys_newfstatat (fs/stat.c:443)
[ 79.811672][  T969] ? lock_is_held_type (kernel/locking/lockdep.c:466 kernel/locking/lockdep.c:5712)
[ 79.812167][  T969] ksys_write (fs/read_write.c:637)
[ 79.812575][  T969] do_syscall_64 (arch/x86/entry/common.c:50 arch/x86/entry/common.c:80)
[ 79.813010][  T969] entry_SYSCALL_64_after_hwframe (arch/x86/entry/entry_64.S:120)
[ 79.813581][  T969] RIP: 0033:0x7f6867fcea37
... SNIP ...
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Executed last

↑

Executed fist

Analysis with Call Trace

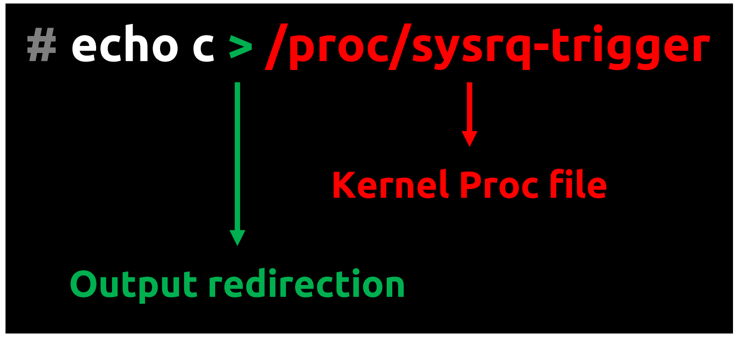
I will analysis except question mark in Call Trace

```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969] dump_stack_lvl (lib/dump_stack.c:107 (discriminator 4))
[ 79.807693][  T969] panic (kernel/panic.c:336)
[ 79.808072][  T969] sysrq_handle_crash (drivers/tty/sysrq.c:155)
[ 79.808583][  T969] __handle_sysrq.cold (drivers/tty/sysrq.c:625)
[ 79.809205][  T969] write_sysrq_trigger (drivers/tty/sysrq.c:1163)
[ 79.809668][  T969] proc_reg_write (./arch/x86/include/asm/atomic.h:165 ./arch/x86/include/asm/atomic.h
:178 ./include/linux/atomic/atomic-arch-fallback.h:611 ./include/linux/atomic/atomic-instrumented.h:266 f
s/proc/inode.c:211 fs/proc/inode.c:353)
[ 79.810195][  T969] ? rcu_read_lock_any_held (kernel/rcu/update.c:347 kernel/rcu/update.c:340)
[ 79.810750][  T969] vfs_write (fs/read_write.c:582)
[ 79.811157][  T969] ? __do_sys_newfstatat (fs/stat.c:443)
[ 79.811672][  T969] ? lock_is_held_type (kernel/locking/lockdep.c:466 kernel/locking/lockdep.c:5712)
[ 79.812167][  T969] ksys_write (fs/read_write.c:637)
[ 79.812575][  T969] do_syscall_64 (arch/x86/entry/common.c:50 arch/x86/entry/common.c:80)
[ 79.813010][  T969] entry_SYSCALL_64_after_hwframe (arch/x86/entry/entry_64.S:120)
[ 79.813581][  T969] RIP: 0033:0x7f6867fcea37
… SNIP …
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

'?' means that the information about this stack entry is probably not reliable.

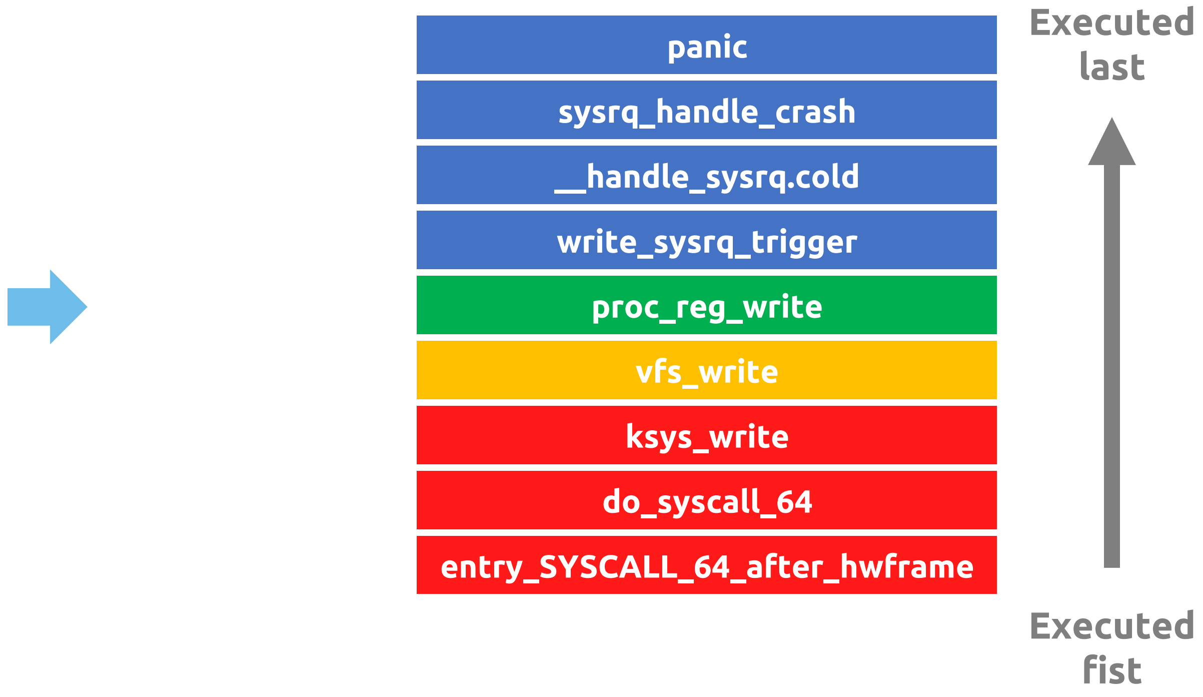
https:///stackoverflow.com/a/13117401

Analysis with Call Trace

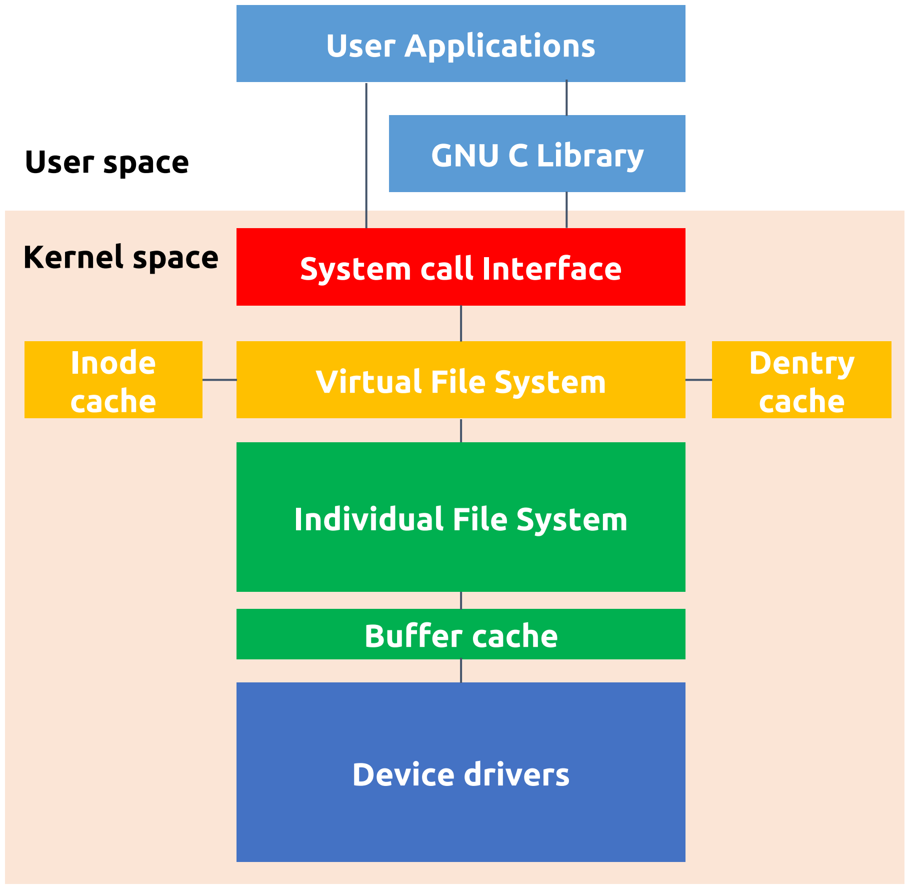


```
# echo c > /proc/sysrq-trigger
```

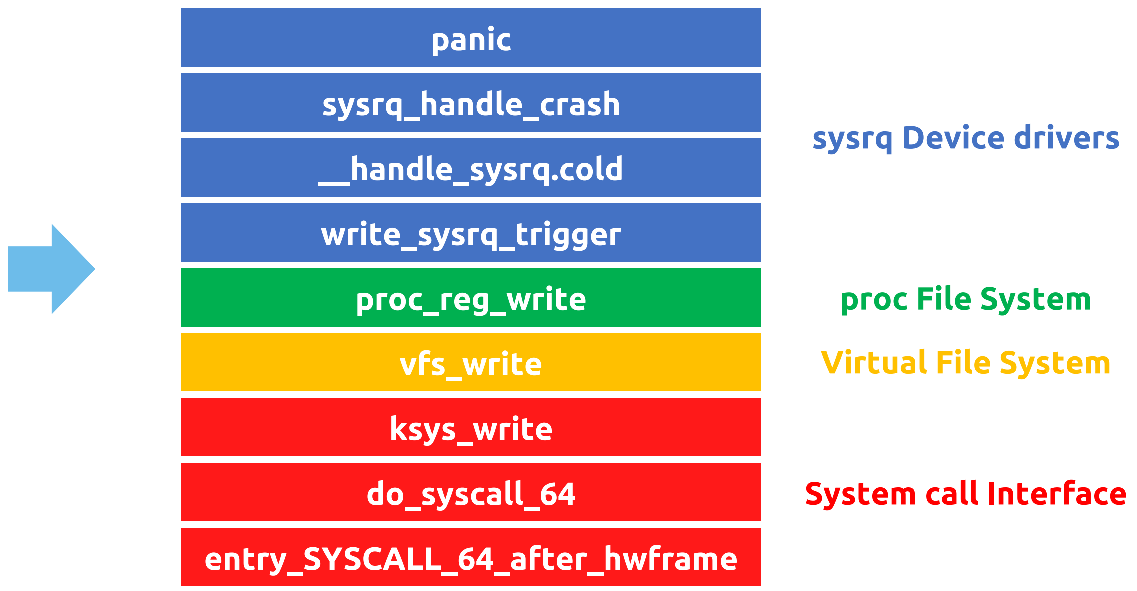
write “c” on /proc/sysrq-trigger



Analysis with Call Trace



https:///developer.ibm.com/tutorials/l-linux-filesystem/



Analysis with Call Trace

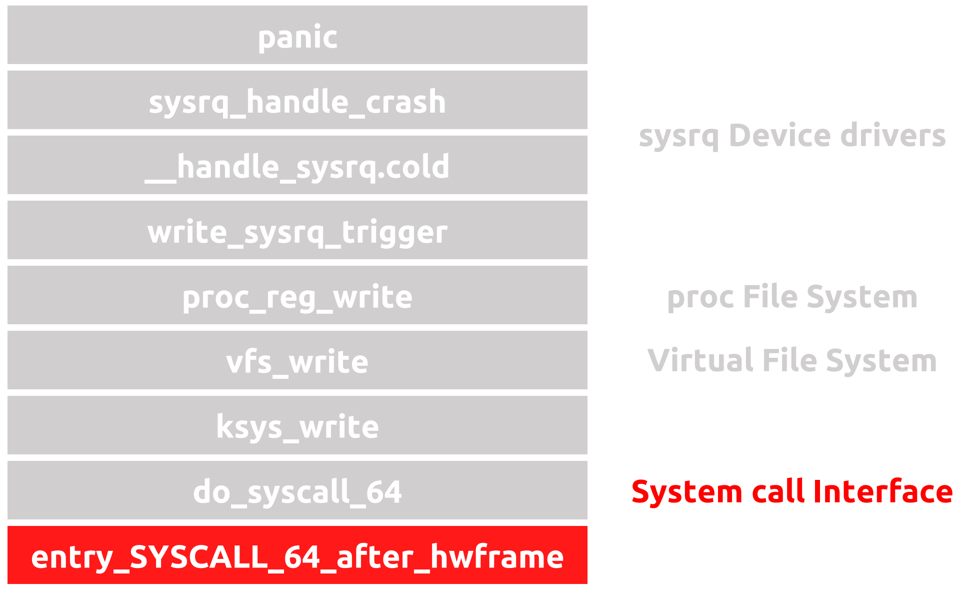
```
[ 79.803379][  T969] sysrq: Trigger a crash
[ 79.803924][  T969] Kernel panic - not syncing: sysrq triggered crash
[ 79.804578][  T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[ 79.805639][  T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
[ 79.806640][  T969] Call Trace:
[ 79.806950][  T969]  <TASK>
[ 79.807229][  T969] dump_stack_lvl (lib/dump_stack.c:107 (discriminator 4))
[ 79.807693][  T969] panic (kernel/panic.c:336)
[ 79.808072][  T969] sysrq_handle_crash (drivers/tty/sysrq.c:155)
[ 79.808583][  T969] __handle_sysrq.cold (drivers/tty/sysrq.c:625)
[ 79.809205][  T969] write_sysrq_trigger (drivers/tty/sysrq.c:1163)
[ 79.809668][  T969] proc_reg_write (./arch/x86/include/asm/atomic.h:165 ./arch/x86/include/asm/atomic.h
:178 ./include/linux/atomic/atomic-arch-fallback.h:611 ./include/linux/atomic/atomic-instrumented.h:266 f
s/proc/inode.c:211 fs/proc/inode.c:353)
[ 79.810195][  T969] ? rcu_read_lock_any_held (kernel/rcu/update.c:347 kernel/rcu/update.c:340)
[ 79.810750][  T969] vfs_write (fs/read_write.c:582)
[ 79.811157][  T969] ? __do_sys_newfstatat (fs/stat.c:443)
[ 79.811672][  T969] ? lock_is_held_type (kernel/locking/lockdep.c:466 kernel/locking/lockdep.c:5712)
[ 79.812167][  T969] ksys_write (fs/read_write.c:637)
[ 79.812575][  T969] do_syscall_64 (arch/x86/entry/common.c:50 arch/x86/entry/common.c:80)
[ 79.813010][  T969] entry_SYSCALL_64_after_hwframe (arch/x86/entry/entry_64.S:120)
[ 79.813581][  T969] RIP: 0033:0x7f6867fcea37
... SNIP ...
[ 79.820728][  T969]  </TASK>
[ 79.821439][  T969] Kernel Offset: disabled
[ 79.821895][  T969] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
```

Let's start using based on 6.2.0-rc2!

entry_SYSCALL_64_after_hwframe

```
SYM_INNER_LABEL(entry_SYSCALL_64_after_hwframe, SYM_L_GLOBAL)
	pushq	%rax					/* pt_regs->orig_ax */

	PUSH_AND_CLEAR_REGS rax=$-ENOSYS

	/* IRQs are off. */
	movq	%rsp, %rdi
	/* Sign extend the lower 32bit as syscall numbers are treated as int */
	movslq	%eax, %rsi

	/* clobbers %rax, make sure it is after saving the syscall nr */
	IBRS_ENTER
	UNTRAIN_RET

	call	do_syscall_64		/* returns with IRQs disabled */

```

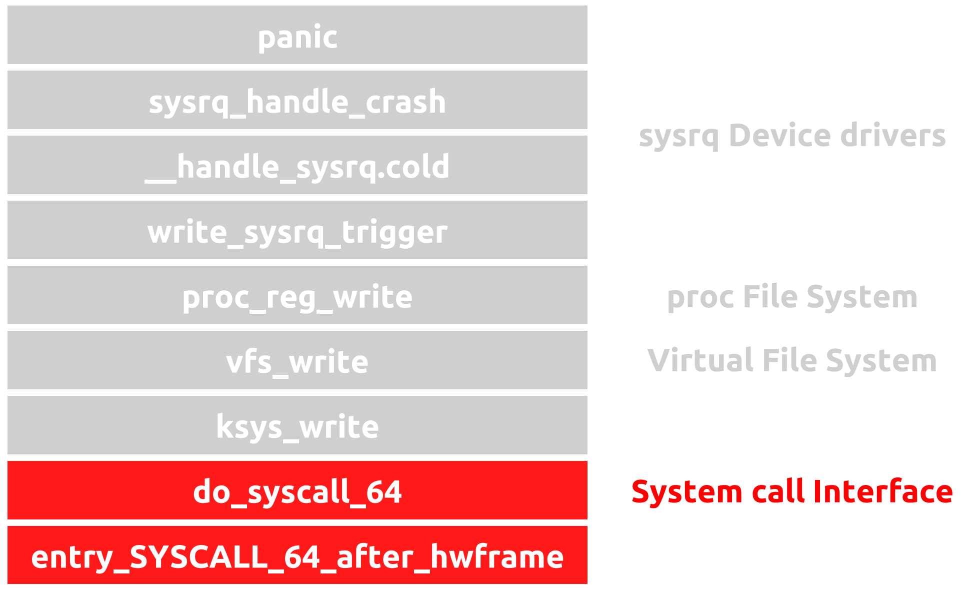
arch/x86/entry/entry_64.S



do_syscall_64

```
__visible noinstr void do_syscall_64(struct pt_regs *regs, int nr)
{
	add_random_kstack_offset();
	nr = syscall_enter_from_user_mode(regs, nr);

	instrumentation_begin();

	if (!do_syscall_x64(regs, nr) && !do_syscall_x32(regs, nr) && nr != -1) {
		/* Invalid system call, but still a system call. */
		regs->ax = __x64_sys_ni_syscall(regs);
	}

	instrumentation_end();
	syscall_exit_to_user_mode(regs);
}
```

arch/x86/entry/common.c

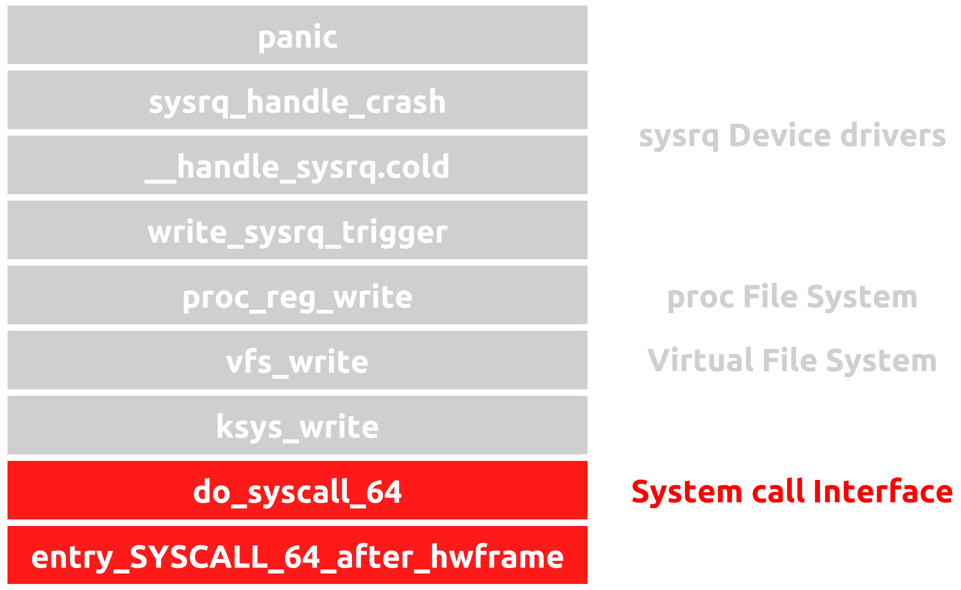


do_syscall_64

```c
__visible noinstr void do_syscall_64(struct pt_regs *regs, int nr)
{
	add_random_kstack_offset();
	nr = syscall_enter_from_user_mode(regs, nr);

	instrumentation_begin();

	if (!do_syscall_x64(regs, nr) && !do_syscall_x32(regs, nr) && nr != -1) {
		/* Invalid system call, but still a system call. */
		regs->ax = __x64_sys_ni_syscall(regs);
	}
```

```c
static __always_inline bool do_syscall_x64(struct pt_regs *regs, int nr)
{
... SNIP ...
	unsigned int unr = nr;

	if (likely(unr < NR_syscalls)) {
		unr = array_index_nospec(unr, NR_syscalls);
		regs->ax = sys_call_table[unr](regs);
		return true;
	}
	return false;
}
```

arch/x86/entry/common.c



do_syscall_64

```
static __always_inline bool do_syscall_x64(struct pt_regs *regs, int nr)
{
... SNIP ...
        regs->ax = sys_call_table[unr](regs);
... SNIP ...
```

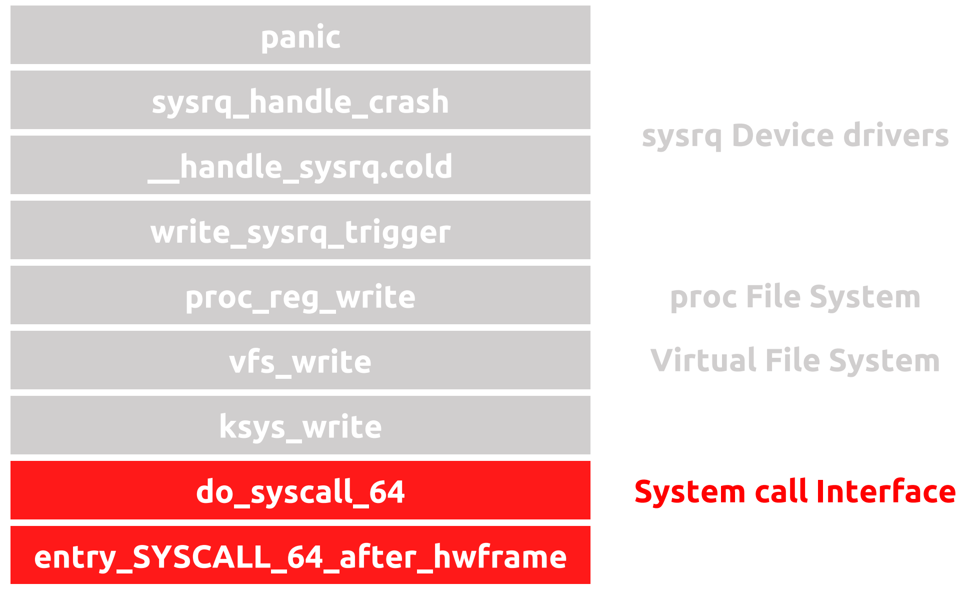
arch/x86/entry/common.c

```
asmlinkage const sys_call_ptr_t sys_call_table[] = {
#include <asm/syscalls_64.h>
};
```

arch/x86/entry/syscall_64.c

```
__SYSCALL(0, sys_read)
__SYSCALL(1, sys_write)
__SYSCALL(2, sys_open)
__SYSCALL(3, sys_close)
__SYSCALL(4, sys_newstat)
__SYSCALL(5, sys_newfstat)
__SYSCALL(6, sys_newlstat)
... SNIP ...
```

arch/x86/include/generated/asm/syscalls_64.h



do_syscall_64

```
static __always_inline bool do_syscall_x64(struct pt_regs *regs, int nr)
{
… SNIP …
        regs->ax = sys_call_table[unr](regs);
… SNIP …
```

arch/x86/entry/common.c

```
asmlinkage const sys_call_ptr_t sys_call_table[] = {
#include <asm/syscalls_64.h>
};
```

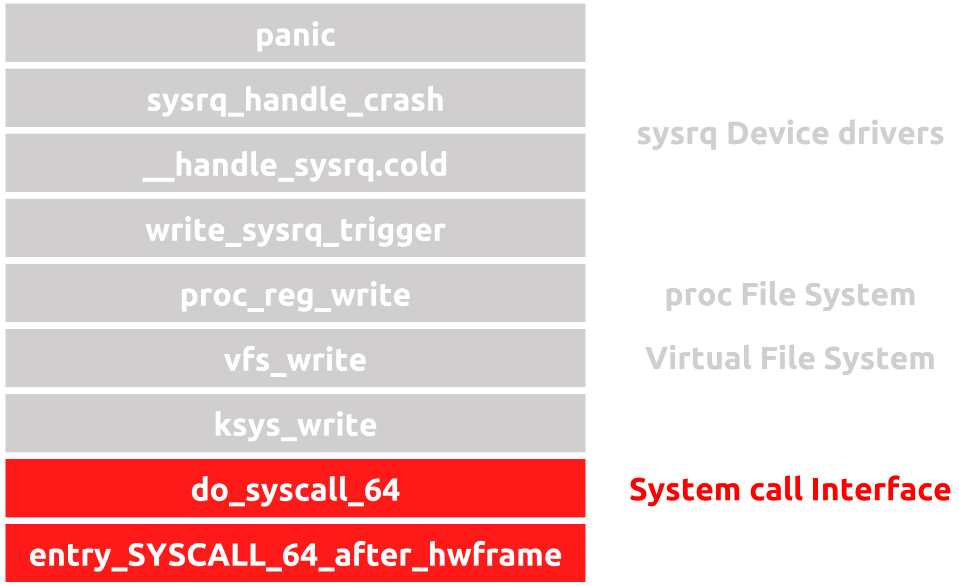
arch/x86/entry/syscall_64.c

```
__SYSCALL(0, sys_read)
__SYSCALL(1, sys_write)
__SYSCALL(2, sys_open)
__SYSCALL(3, sys_close)
__SYSCALL(4, sys_newstat)
__SYSCALL(5, sys_newfstat)
__SYSCALL(6, sys_newlstat)
… SNIP …
```

arch/x86/include/generated/asm/syscalls_64.h

```
SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
                size_t, count)
{
        return ksys_write(fd, buf, count);
}
```

fs/write.c



do_syscall_64

```
__visible noinstr void do_syscall_64(struct pt_regs *regs, int nr)
{
        add_random_kstack_offset();
        nr = syscall_enter_from_user_mode(regs, nr);

        instrumentation_begin();

        if (!do_syscall_x64(regs, nr) && !do_syscall_x32(regs, nr) && nr != -1) {
                /* Invalid system call, but still a system call. */
                regs->ax = __x64_sys_ni_syscall(regs);
        }
```

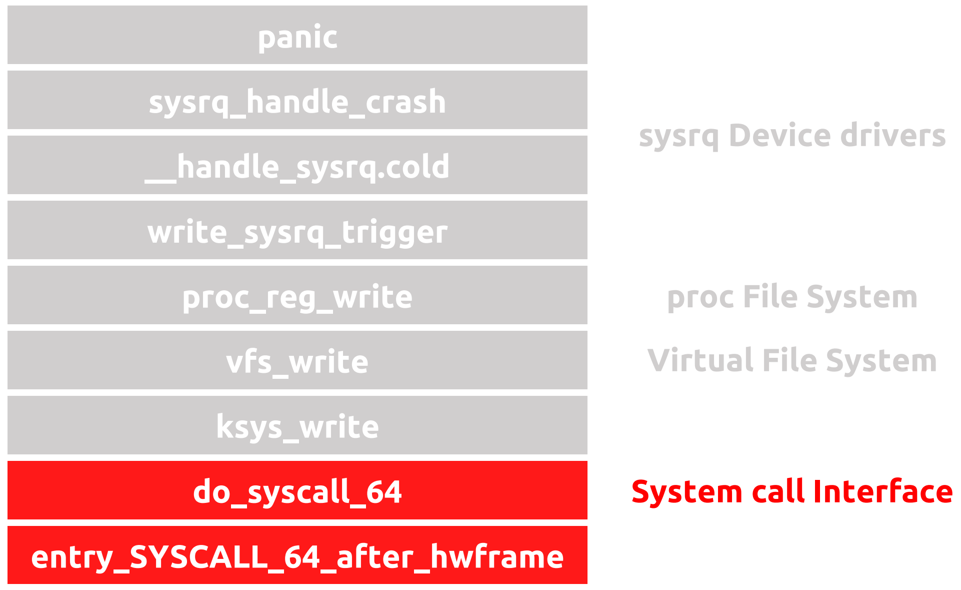
```
static __always_inline bool do_syscall_x64(struct pt_regs *regs, int nr)
{
... SNIP ...
        unsigned int unr = nr;

        if (likely(unr < NR_syscalls)) {
                unr = array_index_nospec(unr, NR_syscalls);
                regs->ax = sys_call_table[unr](regs);
                return true;
        }
        return false;
}
```

↑ calls ksys_write

arch/x86/entry/common.c

```
SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
                size_t, count)
{
        return ksys_write(fd, buf, count);
}
```

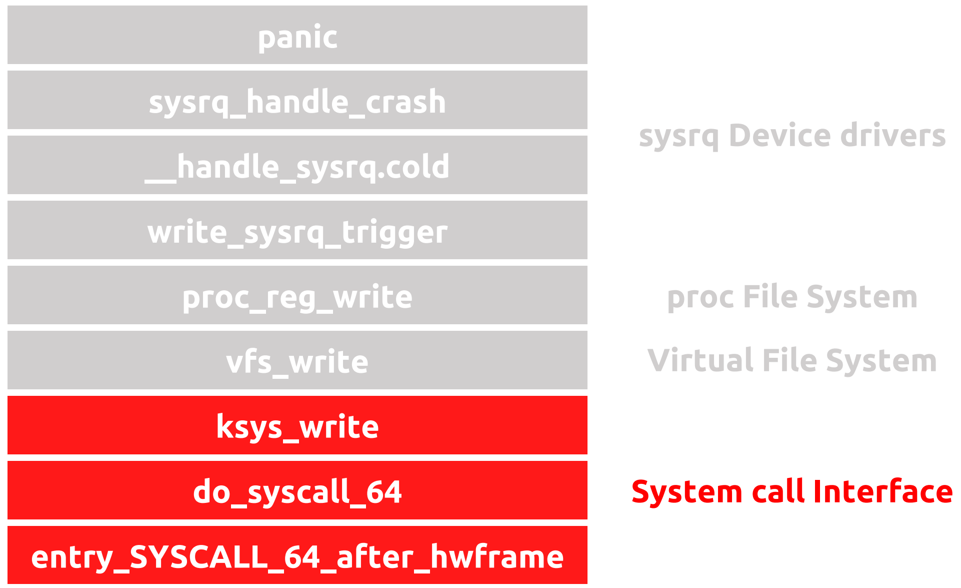
fs/write.c



ksys_write

```
ssize_t ksys_write(unsigned int fd, const char __user *buf, size_t count)
{
        struct fd f = fdget_pos(fd);
        ssize_t ret = -EBADF;

        if (f.file) {
                loff_t pos, *ppos = file_ppos(f.file);
                if (ppos) {
                        pos = *ppos;
                        ppos = &pos;
                }
                ret = vfs_write(f.file, buf, count, ppos);
                if (ret >= 0 && ppos)
                        f.file->f_pos = pos;
                fdput_pos(f);
        }

        return ret;
}
```

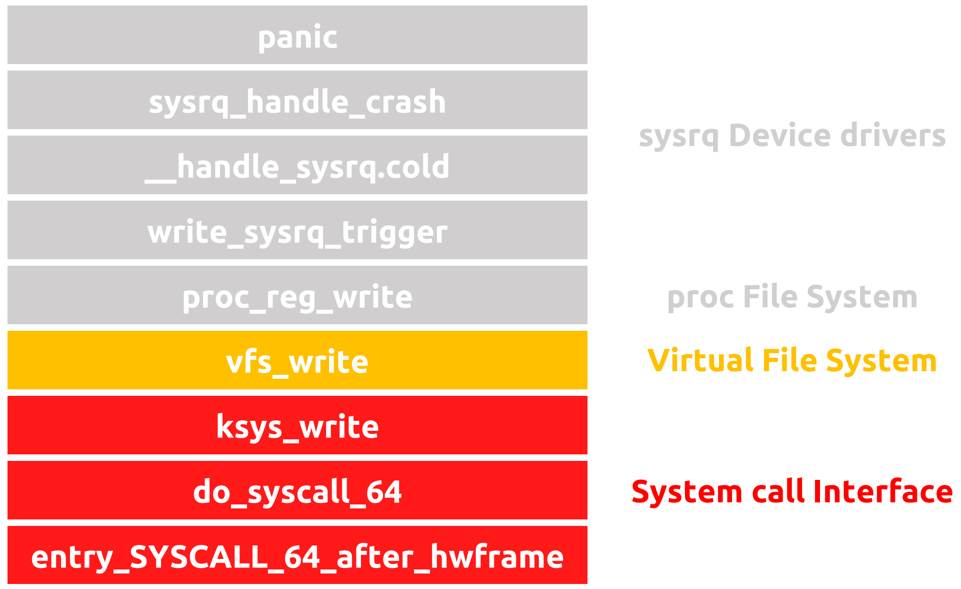
fs/read_write.c



vfs_write

```
ssize_t vfs_write(struct file *file, const char __user *buf, size_t count, loff_t *pos)
{
        ssize_t ret;
… SNIP …
        ret = rw_verify_area(WRITE, file, pos, count);
        if (ret)
                return ret;
        if (count > MAX_RW_COUNT)
                count =  MAX_RW_COUNT;
        file_start_write(file);
        if (file->f_op->write)
                ret = file->f_op->write(file, buf, count, pos);
        else if (file->f_op->write_iter)
                ret = new_sync_write(file, buf, count, pos);
        else
                ret = -EINVAL;
        if (ret > 0) {
                fsnotify_modify(file);
                add_wchar(current, ret);
        }
        inc_syscw(current);
        file_end_write(file);
        return ret;
}
```

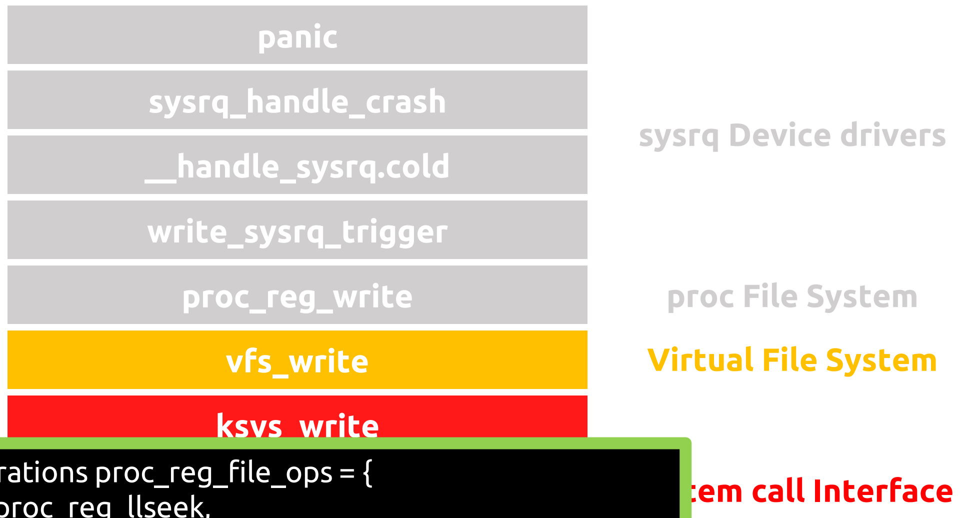
fs/read_write.c



vfs_write

```
ssize_t vfs_write(struct file *file, const char __user *buf, size_t count, loff_t *pos)
{
        ssize_t ret;
... SNIP ...
        ret = rw_verify_area(WRITE, file, pos, count);
        if (ret)
              return ret;
        if (count > MAX_RW_COUNT)
              count =  MAX_RW_COUNT;
        file_start_write(file);
        if (file->f_op->write)                    ↓ calls proc_reg_write
              ret = file->f_op->write(file, buf, count, pos);
        else if (file->f_op->write_iter)
              ret = new_sync_write(file, buf, count, pos);
        else
              ret = -EINVAL;
        if (ret > 0) {
              fsnotify_modify(file);
              add_wchar(current, ret);
        }
        inc_syscw(current);
        file_end_write(file);
        return ret;
}
```

fs/read_write.c



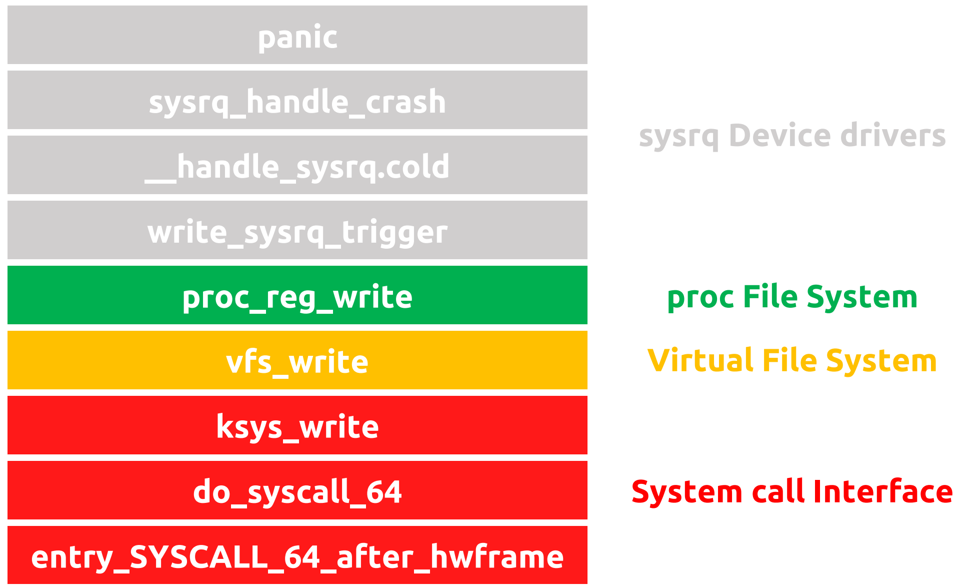
```
static const struct file_operations proc_reg_file_ops = {
        .llseek                 = proc_reg_llseek,
        .read                   = proc_reg_read,
        .write                  = proc_reg_write,
        .poll                   = proc_reg_poll,
        .unlocked_ioctl         = proc_reg_unlocked_ioctl,
        .mmap                   = proc_reg_mmap,
        .get_unmapped_area = proc_reg_get_unmapped_area,
        .open                   = proc_reg_open,
        .release                = proc_reg_release,
};
```

fs/proc/inode.c

proc_reg_write

```c
static ssize_t proc_reg_write(struct file *file, const char __user *buf, size_t count, loff_t *ppos)
{
	struct proc_dir_entry *pde = PDE(file_inode(file));
	ssize_t rv = -EIO;

	if (pde_is_permanent(pde)) {
		return pde_write(pde, file, buf, count, ppos);
	} else if (use_pde(pde)) {
		rv = pde_write(pde, file, buf, count, ppos);
		unuse_pde(pde);
	}
	return rv;
}
```

fs/proc/inode.c

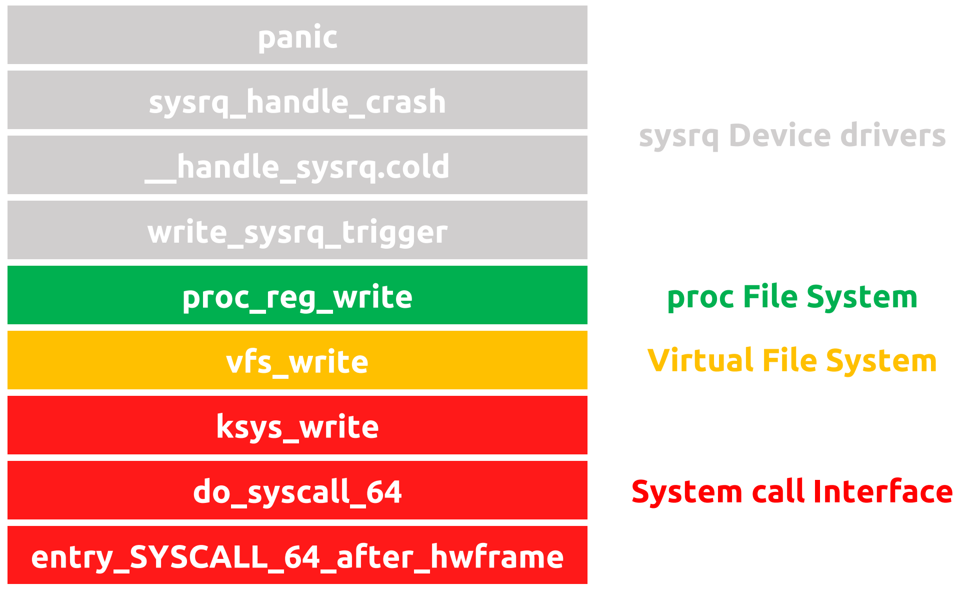


proc_reg_write

```c
static ssize_t proc_reg_write(struct file *file, const char __user *buf, size_t count, loff_t *ppos)
{
        struct proc_dir_entry *pde = PDE(file_inode(file));
        ssize_t rv = -EIO;

        if (pde_is_permanent(pde)) {
                return pde_write(pde, file, buf, count, ppos);
        } else if (use_pde(pde)) {
                rv = pde_write(pde, file, buf, count, ppos);
                unuse_pde(pde);
        }
        return rv;
}
```

```c
static ssize_t pde_write(struct proc_dir_entry *pde, struct file *file, const char __user *buf,
size_t count, loff_t *ppos)
{
        typeof_member(struct proc_ops, proc_write) write;

        write = pde->proc_ops->proc_write;
        if (write)
                return write(file, buf, count, ppos);
        return -EIO;
}
```

fs/proc/inode.c

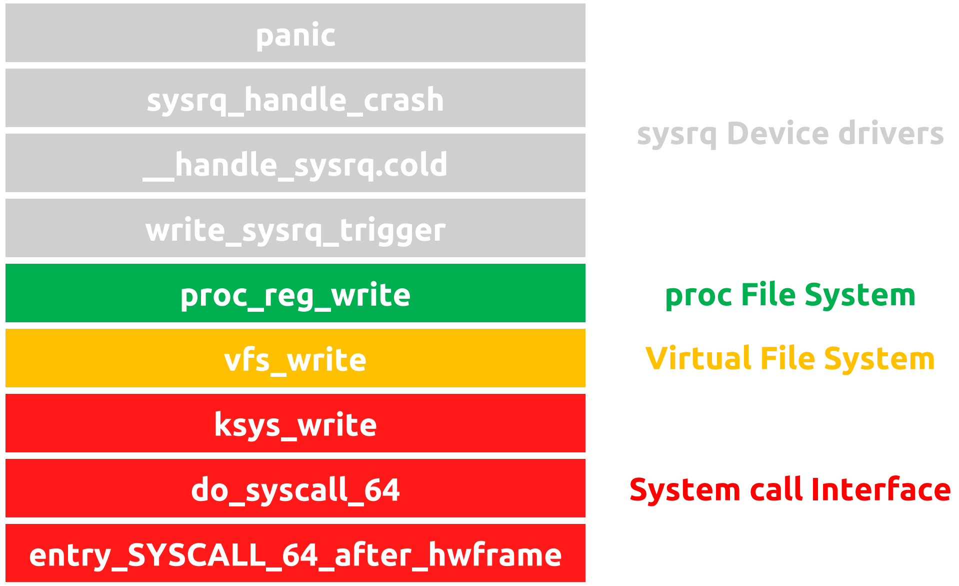


proc_reg_write

```
static ssize_t proc_reg_write(struct file *file, const char __user *buf, size_t count, loff_t *ppos)
{
        struct proc_dir_entry *pde = PDE(file_inode(file));
        ssize_t rv = -EIO;

        if (pde_is_permanent(pde)) {
                return pde_write(pde, file, buf, count, ppos);
        } else if (use_pde(pde)) {
                rv = pde_write(pde, file, buf, count, ppos);
                unuse_pde(pde);
        }
        return rv;
}
```

```
static ssize_t pde_write(struct proc_dir_entry *pde, struct file *file, const char __user *buf,
size_t count, loff_t *ppos)
{
        typeof_member(struct proc_ops, proc_write) write;

        write = pde->proc_ops->proc_write; ← calls write_sysrq_trigger
        if (write)
                return write(file, buf, count, ppos);
        return -EIO;
}
```

fs/proc/inode.c



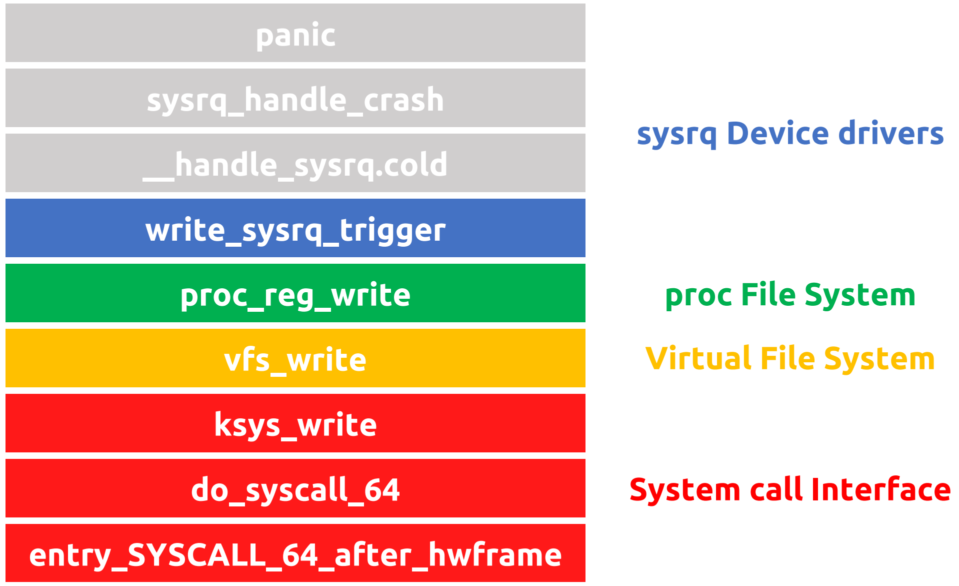
```
static const struct proc_ops sysrq_trigger_proc_ops = {
        .proc_write          = write_sysrq_trigger,
        .proc_lseek          = noop_llseek,
};
```

drivers/tty/sysrq.c

write_sysrq_trigger

```
static ssize_t write_sysrq_trigger(struct file *file, const char __user *buf,
                                   size_t count, loff_t *ppos)
{
        if (count) {
                char c;

                if (get_user(c, buf))
                        return -EFAULT;
                __handle_sysrq(c, false);
        }

        return count;
}
```

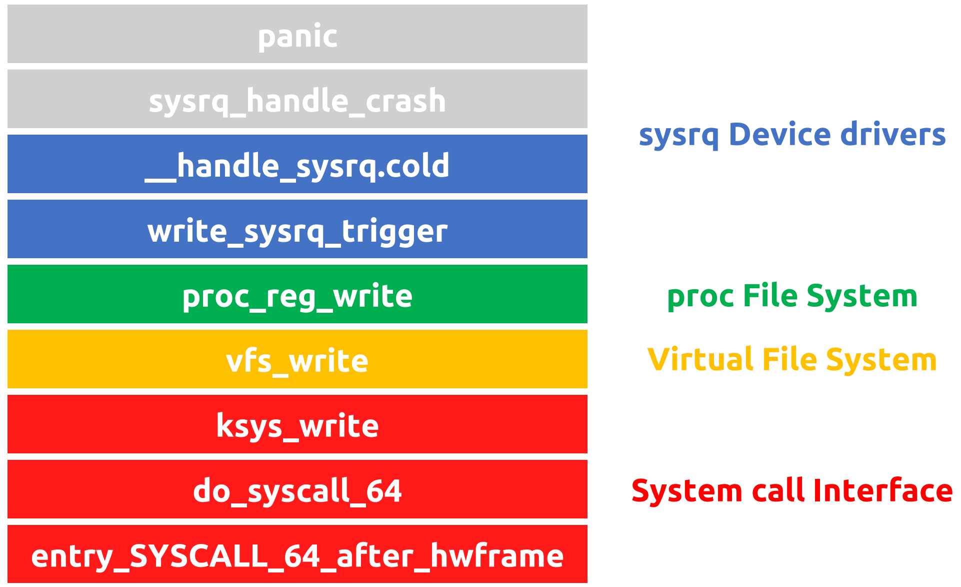
drivers/tty/sysrq.c



__handle_sysrq

```
void __handle_sysrq(int key, bool check_mask)
{
	const struct sysrq_key_op *op_p;
	int orig_log_level;
	int orig_suppress_printk;
	int i;
... SNIP ...
	op_p = __sysrq_get_key_op(key);
	if (op_p) {
		/*
		 * Should we check for enabled operations (/proc/sysrq-trigger
		 * should not) and is the invoked operation enabled?
		 */
		if (!check_mask || sysrq_on_mask(op_p->enable_mask)) {
			pr_info("%s\n", op_p->action_msg);
			console_loglevel = orig_log_level;
			op_p->handler(key);
		} else {
			pr_info("This sysrq operation is disabled.\n");
			console_loglevel = orig_log_level;
		}
```

drivers/tty/sysrq.c

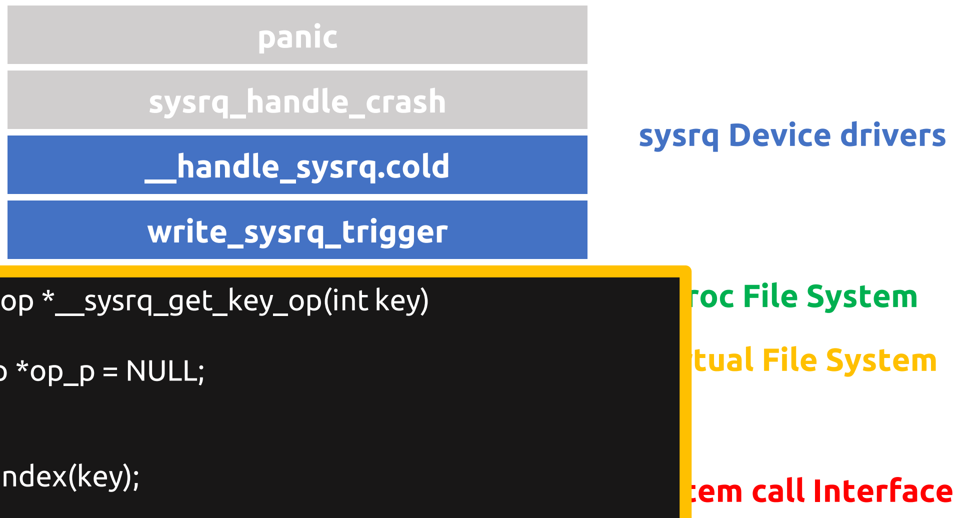


__handle_sysrq

```
void __handle_sysrq(int key, bool check_mask)
{
        const struct sysrq_key_op *op_p;
        int orig_log_level;
        int orig_suppress_printk;
        int i;
... SNIP ...
        op_p = __sysrq_get_key_op(key);
        if (op_p) {
                /*
                 * Should we check for enabled operations (/proc/sysrq-trigger
                 * should not) and is the invoked operation enabled?
                 */
                if (!check_mask || sysrq_on_mask(op_p->enable_mask)) {
                        pr_info("%s\n", op_p->action_msg);
                        console_loglevel = orig_log_level;
                        op_p->handler(key);
                } else {
                        pr_info("This sysrq operation is disabled.\n");
                        console_loglevel = orig_log_level;
                }
```

drivers/tty/sysrq.c

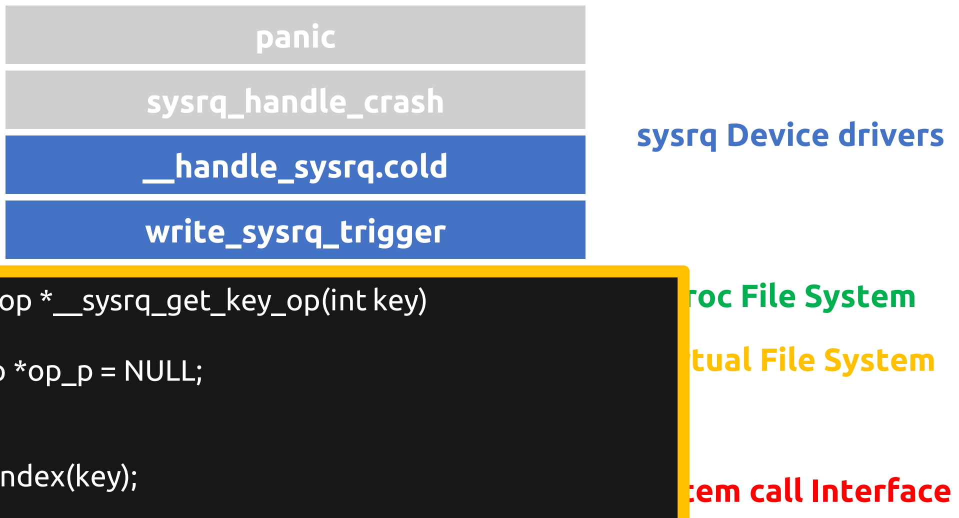
```
static const struct sysrq_key_op *__sysrq_get_key_op(int key)
{
        const struct sysrq_key_op *op_p = NULL;
        int i;

        i = sysrq_key_table_key2index(key);
        if (i != -1)
                op_p = sysrq_key_table[i];

        return op_p;
}
```



__handle_sysrq

```c
void __handle_sysrq(int key, bool check_mask)
{
        const struct sysrq_key_op *op_p;
        int orig_log_level;
        int orig_suppress_printk;
        int i;
… SNIP …
        op_p = __sysrq_get_key_op(key);
        if (op_p) {
                /*
                 * Should we check for enabled operations (/proc/sysrq-trigger
                 * should not) and is the invoked operation enabled?
                 */
                if (!check_mask || sysrq_on_mask(op_p->enable_mask)) {
                        pr_info("%s\n", op_p->action_msg);
                        console_loglevel = orig_log_level;
                        op_p->handler(key);
                } else {
                        pr_info("This sysrq operation is disabled.\n");
                        console_loglevel = orig_log_level;
                }
```

drivers/tty/sysrq.c



```c
static const struct sysrq_key_op *__sysrq_get_key_op(int key)
{
        const struct sysrq_key_op *op_p = NULL;
        int i;

        i = sysrq_key_table_key2index(key);
        if (i != -1)
                op_p = sysrq_key_table[i];

        return op_p;
}
```

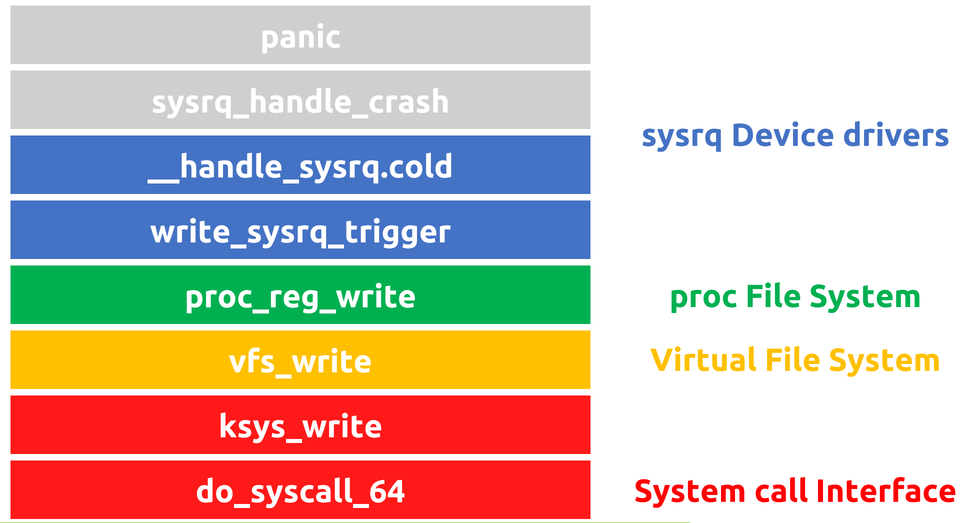
```c
static const struct sysrq_key_op *sysrq_key_table[62] = {
…
        &sysrq_crash_op,                        /* c */
        &sysrq_showlocks_op,                    /* d */
        &sysrq_term_op,                         /* e */
        &sysrq_moom_op,                         /* f */
        /* g: May be registered for the kernel debugger */
```

__handle_sysrq

```
void __handle_sysrq(int key, bool check_mask)
{
        const struct sysrq_key_op *op_p;
        int orig_log_level;
        int orig_suppress_printk;
        int i;
... SNIP ...
        op_p = __sysrq_get_key_op(key);
        if (op_p) {
                /*
                 * Should we check for enabled operations (/proc/sysrq-trigger
                 * should not) and is the invoked operation enabled?
                 */
                if (!check_mask || sysrq_on_mask(op_p->enable_mask)) {
                        pr_info("%s\n", op_p->action_msg);
                        console_loglevel = orig_log_level;
                        op_p->handler(key); ← calls sysrq_handle_crash
                } else {
                        pr_info("This sysrq operation is disabled.\n");
                        console_loglevel = orig_log_level;
                }
```

drivers/tty/sysrq.c

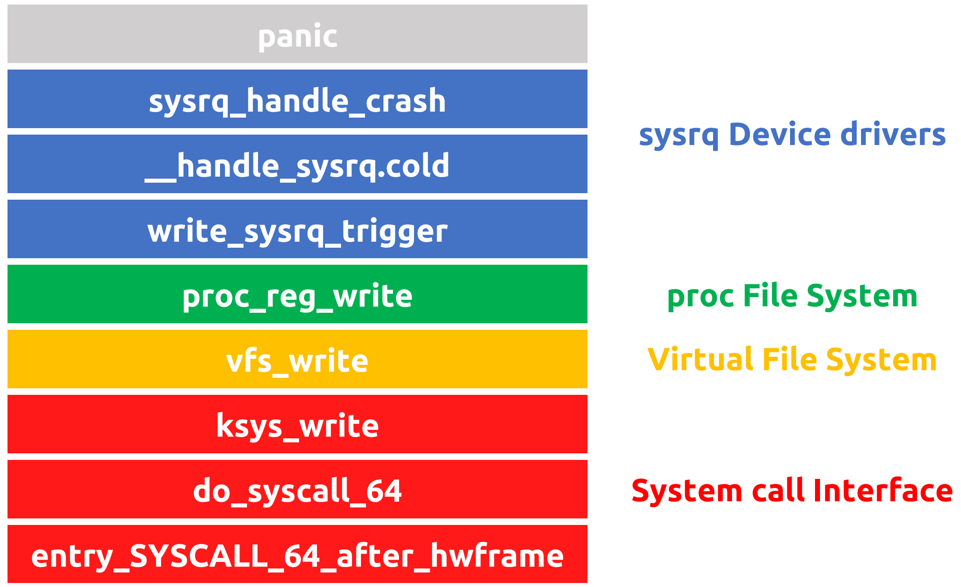
```
static const struct sysrq_key_op sysrq_crash_op = {
        .handler                = sysrq_handle_crash,
        .help_msg               = "crash(c)",
        .action_msg             = "Trigger a crash",
        .enable_mask            = SYSRQ_ENABLE_DUMP,
};
```



sysrq_handle_crash

```c
static void sysrq_handle_crash(int key)
{
	/* release the RCU read lock before crashing */
	rcu_read_unlock();

	panic("sysrq triggered crash\n");
}
```

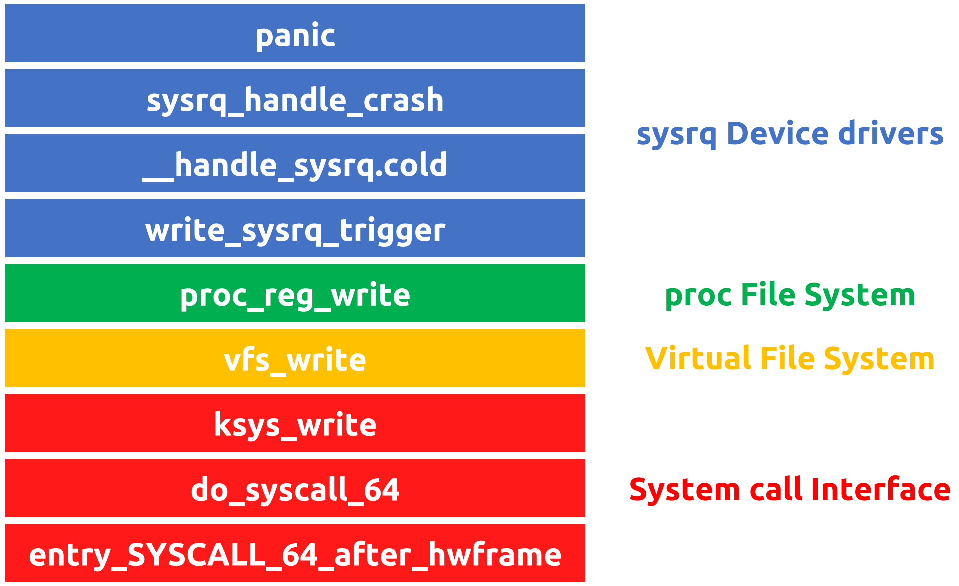
drivers/tty/sysrq.c



panic

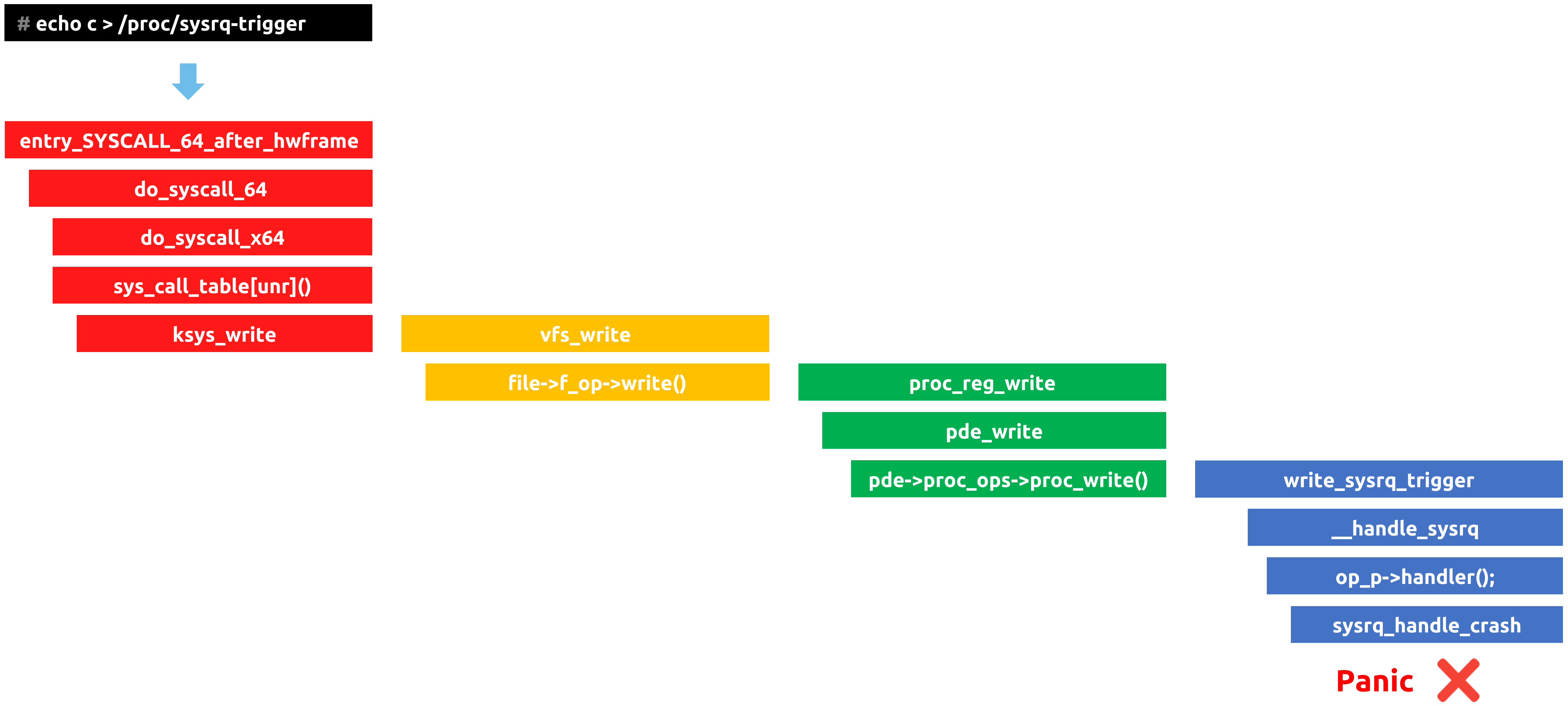
```
static void sysrq_handle_crash(int key)
{
        /* release the RCU read lock before crashing */
        rcu_read_unlock();

        panic("sysrq triggered crash\n");
}
```

drivers/tty/sysrq.c



Summary

```
# echo c > /proc/sysrq-trigger
```



Analysis kernel panic log

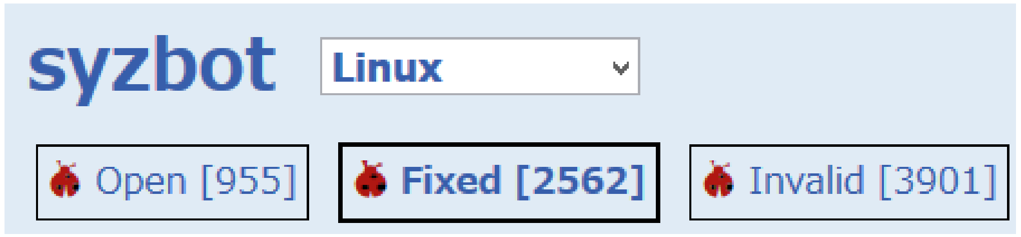
① Skim through Kernel Panic log

② Decode call trace

③ In depth analysis

④ **Additional resources**

- Tracking the syzbot dashboard



https:////syzkaller.appspot.com/upstream

syzbot

Information about the bug

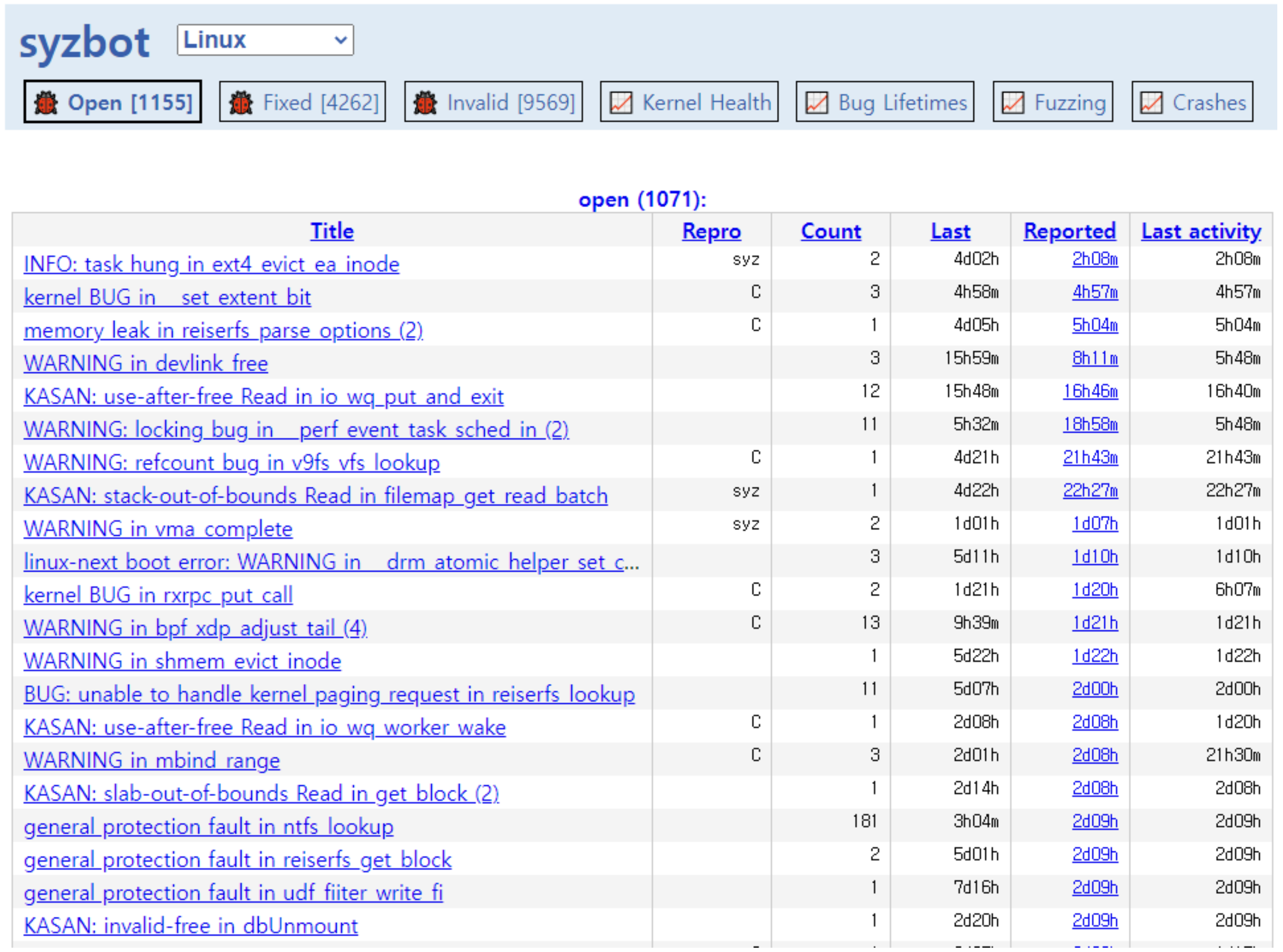
when it started happening

when it last happened

how frequently

more crash reports

etc..



open (1071):

Title	Repro	Count	Last	Reported	Last activity
INFO: task hung in ext4_evict_ea_inode	syz	2	4d02h	2h08m	2h08m
kernel BUG in __set_extent_bit	C	3	4h58m	4h57m	4h57m
memory leak in reiserfs_parse_options (2)	C	1	4d05h	5h04m	5h04m
WARNING in devlink_free		3	15h59m	8h11m	5h48m
KASAN: use-after-free Read in io_wq_put_and_exit		12	15h48m	16h46m	16h40m
WARNING: locking bug in __perf_event_task_sched_in (2)		11	5h32m	18h58m	5h48m
WARNING: refcount bug in v9fs_vfs_lookup	C	1	4d21h	21h43m	21h43m
KASAN: stack-out-of-bounds Read in filemap_get_read_batch	syz	1	4d22h	22h27m	22h27m
WARNING in vma_complete	syz	2	1d01h	1d07h	1d01h
linux-next boot error: WARNING in __drm_atomic_helper_set_c...		3	5d11h	1d10h	1d10h
kernel BUG in rxrpc_put_call	C	2	1d21h	1d20h	6h07m
WARNING in bpf_xdp_adjust_tail (4)	C	13	9h39m	1d21h	1d21h
WARNING in shmem_evict_inode		1	5d22h	1d22h	1d22h
BUG: unable to handle kernel paging request in reiserfs_lookup		11	5d07h	2d00h	2d00h
KASAN: use-after-free Read in io_wq_worker_wake	C	1	2d08h	2d08h	1d20h
WARNING in mbind_range	C	3	2d01h	2d08h	21h30m
KASAN: slab-out-of-bounds Read in get_block (2)		1	2d14h	2d08h	2d08h
general protection fault in ntfs_lookup		181	3h04m	2d09h	2d09h
general protection fault in reiserfs_get_block		2	5d01h	2d09h	2d09h
general protection fault in udf_fiiter_write_fi		1	7d16h	2d09h	2d09h
KASAN: invalid-free in dbUnmount		1	2d20h	2d09h	2d09h

With troubleshooting kernel panic,

I could improve my contribution skills!

THE LINUX FOUNDATION

Q & A

Juhee Kang(claudiajkang@gmail.com)