

Leveling Up My Linux Kernel Contributions

- Troubleshooting the kernel panic

Juhee Kang





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

```
. . .
                                     QEMU
    4.2518591 driver: sr
    4.2520461 Kernel panic - not syncing: UFS: Unable to mount root fs on unkn
    4.2523911 CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.12.3 #1
    4.3022831 Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS rel-
11.1-0-g0551a4be2c-prebuilt.qemu-project.org 04/01/2014
    4.3026311 Call Trace:
    4.3038231 dump_stack+0x4d/0x70
      .3039911 panic+0xca/0x203
     4.3041541 ? printk+0x3e/0x46
    4.3043051 mount_block_root+0x171/0x221
    4.304401] ? set_debug_rodata+0x12/0x12
    4.3044961 mount_root+0x101/0x10a
     4.3045901 ? set_debug_rodata+0x12/0x12
     4.3046391 prepare_namespace+0x134/0x160
    4.3046851 kernel_init_freeable+0x1da/0x1f3
    4.3047301 ? rest_init+0x80/0x80
    4.3047751 kernel_init+0x9/0x100
    4.3048181 ret_from_fork+0x22/0x30
    4.3053031 Kernel Offset: 0x37200000 from 0xffffffff81000000 (relocation re
   0xfffffffff80000000-0xfffffffffffffffff
    4.3056321 --- [ end Kernel panic - not syncing: VFS: Unable to mount root f
 n unknown-block(0,0)
```



```
[79.803379][ T969] sysrg: Trigger a crash
79.803924][ T969] Kernel panic - not syncing: sysrg 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 sysrg 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: ffffffffffffda RBX: 00000000000002 RCX: 00007f6867fcea37
. 79.817605][ T969] RDX: 000000000000002 RSI: 000055cccc98b330 RDI: 000000000000000
79.818378][ T969] RBP: 000055cccc98b330 R08: 00000000000000 R09: 000055cccc98b330
` 79.819154][ T969] R10: 00007f68680d3d60 R11: 000000000000246 R12: 00000000000000
` 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: sysrg 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
            T969] Call Trace:
79.806640][
 79.8069501
            T969] <TASK>
            T969] dump stack lvl+0x49/0x61
 79.807229][
            T969] panic+0x113/0x28c
            T969] sysrq handle crash+0x15/0x20
 79.8080721
79.808583][ T969] handle sysrq.cold+0v51/0v12c
79.809205]
            T969] write sysrq trigger
            T969] proc_reg_write+0x.
 79.8096681
            T969]?rcu read lock anv
 79.810195
            T969] vfs write+0xc7/0x4
            T969]? do sys newfsta
[79.811672][ T969] ?lock is held type
            T969] ksys write+0x67/0x
                                            Panic
            [ T969] do_syscall_64+0x3f
79.813010][ T969] entry SYSCALL 64
            T969] RIP: 0033:0x7f6867
[ 79.814029][ T969] Code: 10 00 f7 d8 64
                                                             b7 0f 1f 00 f3 0f 1e fa 64 8b 0
                                      ... 77 51 c3 48 83 ec 28 48 89 54 ...
4 25 18 00 00 00 85 c0 75 10 b8 01 00 00 5
[79.816012][T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG RAX: 0000000000000001
79.816832][ T969] RAX: ffffffffffffda RBX: 000000000000002 RCX: 00007f6867fcea37
79.817605][ T969] RDX: 000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][ T969] RBP: 000055cccc98b330 R08: 000000000000000 <u>R09: 000055cccc98b330</u>
79.819154][ T969] R10: 00007f68680d3d60 R11: 000000000000246 R12: 00000000000000
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: sysrg triggered crash ]---
```

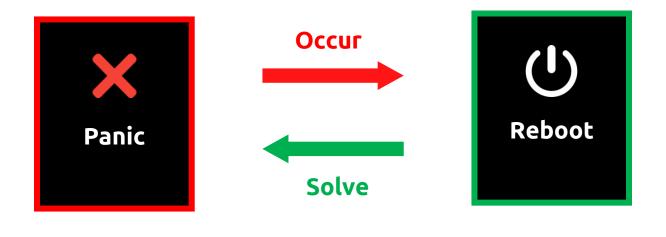


Normally I solved the kernel panic

```
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
           T969] Call Trace:
 79.806640][
 79.8069501
           T969] <TASK>
           T969] dump stack lvl+0x49/0x61
 79.807229][
           T969] panic+0x113/0x28c
 79.807693]
           T969] sysrq handle crash+0x15/0x20
 79.8080721
79.808583][ T969] __handle_sysrq.cold+0v51/0v12c
79.809205]
           T969] write sysrq trigger
           T969] proc_reg_write+0x!
 79.8096681
           T969]?rcu read lock any
 79.810195
           T969] vfs write+0xc7/0x4
           T969] ? __do_sys_newfsta
79.811672][ T969] ?lock is held type
           T969] ksys write+0x67/0>
                                          Just
           T969] do_syscall_64+0x3f
79.813010][ T969] entry SYSCALL 64
                                        Reboot
           T969] RIP: 0033:0x7f68671
[ 79.814029][ T969] Code: 10 00 f7 d8 64
                                                         b7 0f 1f 00 f3 0f 1e fa 64 8b 0
[79.816012][T969] RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG RAX: 0000000000000001
. 79.816832][ T969] RAX: ffffffffffffda RBX: 000000000000002 RCX: 00007f6867fcea37
79.817605][ T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 0000000000000001
[ 79.818378][ T969] RBP: 000055cccc98b330 R08: 00000000000000 R09: 000055cccc98b330
79.819154][ T969] R10: 00007f68680d3d60 R11: 000000000000246 R12: 00000000000000
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: sysrg triggered crash ]---
```



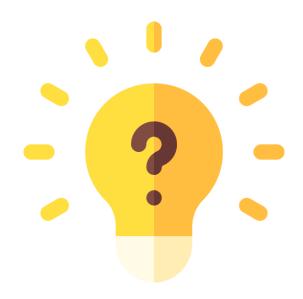
Normally I solved the kernel panic



Infinite loop...



Troubleshooting kernel panic?



Is there any effective way of troubleshooting kernel panic?



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] sysrg: Trigger a crash
[ 79.803924][ T969] Kernel panic - not syncing: sysrg 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 sysrg 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+0x<u>72/0xdc</u>
[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: 000000000000001
 79.816832][ T969] RAX: ffffffffffffda RBX: 000000000000002 RCX: 00007f6867fcea37
 [ 79.817605][ T969] RDX: 000000000000002 RSI: 000055cccc98b330 RDI: 000000000000000
[79.818378][ T969] RBP: 000055cccc98b330 R08: 00000000000000 R09: 000055cccc98b330
 [ 79.819154][ T969] R10: 00007f68680d3d60 R11: 000000000000246 R12: 00000000000000
 [ 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





First, let's look at the Kernel Panic log

① Skim through Kernel Panic log

```
79.803379][ T969] sysrg: Trigger a crash
[ 79.803924][ T969] Kernel panic - not syncing: sysrg 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 sysrg 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: 000000000000001
79.816832][ T969] RAX: ffffffffffffda RBX: 000000000000002 RCX: 00007f6867fcea37
. 79.817605][ T969] RDX: 000000000000002 RSI: 000055cccc98b330 RDI: 000000000000000
[79.818378][ T969] RBP: 000055cccc98b330 R08: 000000000000000 R09: 000055cccc98b330
` 79.819154][ T969] R10: 00007f68680d3d60 R11: 000000000000246 R12: 00000000000000
` 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: sysrg triggered crash ]---
```

① Skim through Kernel Panic log

Panic log header provides the abstraction about the crash

ex) Cause of the panic

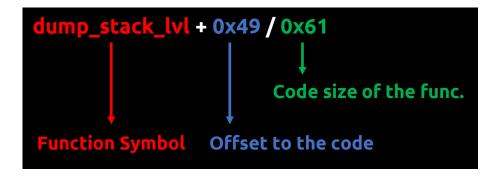
```
79.803379][ T969
                  Kernel panic - not syncing: sysrq triggered crash
[79.803924][ T969
                  CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
[79.804578][ T969
                  Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
<sup>⁻</sup>79.805639][ T969
79.806640][ T969<mark>| call frace.</mark>
79.806950][ T969] <TASK>
                                                 Panic log header
[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: ffffffffffffda RBX: 00000000000002 RCX: 00007f6867fcea37
[ 79.817605][ T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 000000000000000
[ 79.818378][ T969] RBP: 000055cccc98b330 R08: 000000000000000 R09: 000055cccc98b330
[ 79.819154][ T969] R10: 00007f68680d3d60 R11: 000000000000246 R12: 000000000000000
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 ]---
```



① Skim through Kernel Panic log

Panic log header provides the abstraction about the crash

Call Trace provides the context information about the crash



```
T969] Kernel panic - not syncing: sysrq triggered crash
            T9691 CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
                                                                , 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
                  panic+0x113/0x28c
79.807693][ T969
                  sysrg handle crash+0x15/0x20
79.808072][ T969
                  handle sysrq.cold+0x51/0x13c
<sup>-</sup> 79.808583∏ T969
                  write sysrq trigger+0x42/0x50
79.809205][ T969
                  proc reg write+0x54/0xa0
                                                                           Call Trace
<sup>™</sup>79.8096681 T969
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
                  do syscall 64+0x3f/0x90
79.812575][ T969
                  entry SYSCALL 64 after hwframe+0x72/0xdc
<sup>-</sup> 79.813010∏ T969
[79.813581][ T969<mark>[RIP. 0055.0X/1080/TCea5</mark>7
[ 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: ffffffffffffda RBX: 000000000000002 RCX: 00007f6867fcea37
[ 79.817605][ T969] RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 000000000000000
79.818378][ T969] RBP: 000055cccc98b330 R08: 00000000000000 R09: 000055cccc98b330
79.819154][ T969] R10: 00007f68680d3d60 R11: 000000000000246 R12: 00000000000000
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 ]---
```



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

```
79.803924][ T969] Kernel panic - not syncing: sysrg 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
            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
            T969] ksys write+0x67/0xf0
                                                                      Register Info
 79.812575][ T969] do svscall 64+0x3f/0x90
                  RIP: 0033:0x7f6867fcea37
「79.813581∏ 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
[79.814029][ T969
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 ...
                 RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG RAX: 0000000000000001
[ 79.816012][ T969
                  RAX: ffffffffffffda RBX: 000000000000002 RCX: 00007f6867fcea37
[79.816832][ T969
79.817605][ T969 RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 000000000000000
                  RBP: 000055cccc98b330 R08: 00000000000000 R09: 000055cccc98b330
                 R10: 00007f68680d3d60 R11: 000000000000246 R12: 000000000000002
` 79.819935][ T969<mark>'</mark> R13: 00007f68680d4780 R14: 00007f68680d0600 R15: 00007f68680cfa00
[79.820728][ T969] </r>
79.821439][ T969] Kernel Offset: disabled
79.821895][ T969] ---[ end Kernel panic - not syncing: sysrg triggered crash ]---
```



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

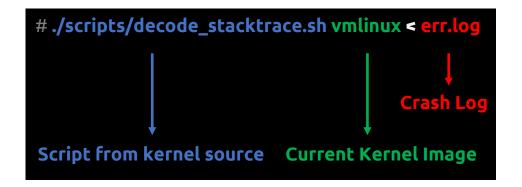
```
79.803924][ T969] Kernel panic - not syncing: sysrg triggered crash
79.804578][ T969] CPU: 6 PID: 969 Comm: bash Not tainted 6.2.0-rc2 #2 e5c88a38d28ce676364b6ebf0...
79.805639][ T969] Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 1.15.0-1 04/01/2014
79.806640][ T969] Call Trace:
 79.806950][ T969] <TASK>
79.807229][ T969] dump stack lvl+0x49/0x61
            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
            T969] ksys write+0x67/0xf0
                                                                      Register Info
 79.812575][ T969] do svscall 64+0x3f/0x90
[79.813581][ T969
                  RIP: 0033:0x7f6867fcea37
ີ 79.814029ໄΓ 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 ...
                 RSP: 002b:00007ffcd75d2a18 EFLAGS: 00000246 ORIG RAX: 0000000000000001
「79.816012]「T969
<u>79.8</u>16832∏ T969
                  RAX: ffffffffffffda RBX: 00000000000002 RCX: 00007f6867fcea37
79.817605][ T969 RDX: 0000000000000002 RSI: 000055cccc98b330 RDI: 000000000000000
                  RBP: 000055cccc98b330 R08: 00000000000000 R09: 000055cccc98b330
[79.818378][ T969
                 R10: 00007f68680d3d60 R11: 000000000000246 R12: 000000000000002
 79.819154][ T969
79.819935][ T969<mark> R13</mark>: 00007f68680d4780 R14: 00007f68680d0600 <u>R15: 00007f68680cfa00</u>
79.820728][ T969] </1A3N>
79.821439][ T969] Kernel Offset: disabled
79.821895][ T969] ---[ end Kernel panic - not syncing: sysrg triggered crash ]---
```

① Skim through Kernel Panic log

For call trace analysis, save the panic log



- ① Skim through Kernel Panic log
- ② Decode call trace





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] sysrg: 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>
[ 79.807693][ T969] panic (
[79.808072][ T969] sysrq handle crash (
79.808583][ T969] __handle_sysrq.cold (
[79.809205][ T969] write sysrq trigger (
79.809668][ T969] proc reg write (.
79.810195][ T969]?rcu_read_lock_any_held(
[79.810750][ T969] vfs write (fs
79.811157][ T969]? do sys newfstatat (
[79.811672][ T969]?lock_is_held_type (
[ 79.812167][ T969] ksys write (f
[79.812575][ T969] do syscall 64 (a
```

Shows the code information of a symbol



Let's take a deep dive into

how kernel panic actually occurs!



- ① Skim through Kernel Panic log
- ② Decode call trace
- ③ In depth analysis
 - Panic trigger command

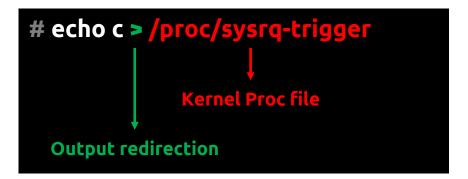


- ① Skim through Kernel Panic log
- ② Decode call trace
- ③ In depth analysis
 - Panic trigger command

echo c > /proc/sysrq-trigger

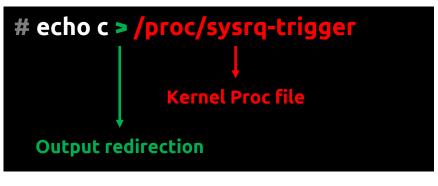


- **1** Skim through Kernel Panic log
- ② Decode call trace
- ③ In depth analysis
 - Panic trigger command





- ① Skim through Kernel Panic log
- ② Decode call trace
- ③ In depth analysis
 - Panic trigger command





write "c" on /proc/sysrq-trigger

- **1** Skim through Kernel Panic log
- ② Decode call trace
- ③ In depth analysis
 - Panic trigger command
 - Analysis with Call Trace

- ① Skim through Kernel Panic log
- ② Decode call trace
- ③ In depth analysis
 - Panic trigger command
 - Analysis with Call Trace

```
79.803379][ T969] sysrg: Trigger a crash
 79.803924][ T969] Kernel panic - not syncing: sysrg 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)
79.809205][ 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)
 [79.813581][ 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

fist



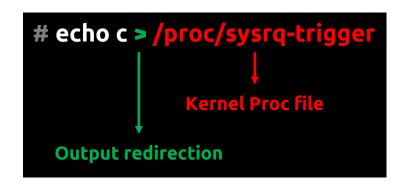
I will analysis except question mark in Call Trace

```
79.803379][ T969] sysrq: Trigger a crash
79.803924][ T969] Kernel panic - not syncing: sysrg 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]
79.807693][ T969]
79.808072][ T969]
79.808583][ T969<sup>-</sup>
79.809205][ T969<sup>-</sup>
`79.809668][ T969]
[79.810195][ T969] ? rcu_read_lock_any_held (kernel/rcu/update.c:347 kernel/rcu/update.c:340)
79.810750][ T969]
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]
79.812575][ T969]
79.813010][ T969] (
[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 ]---
```

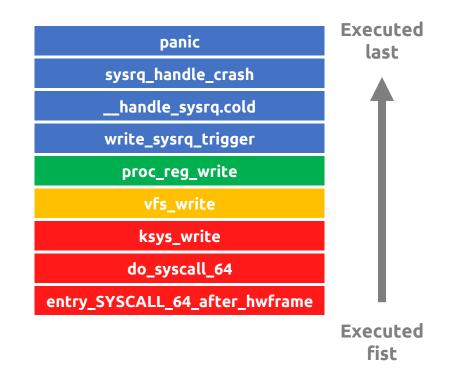
'?' means that the information about this stack entry is probably not reliable.

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



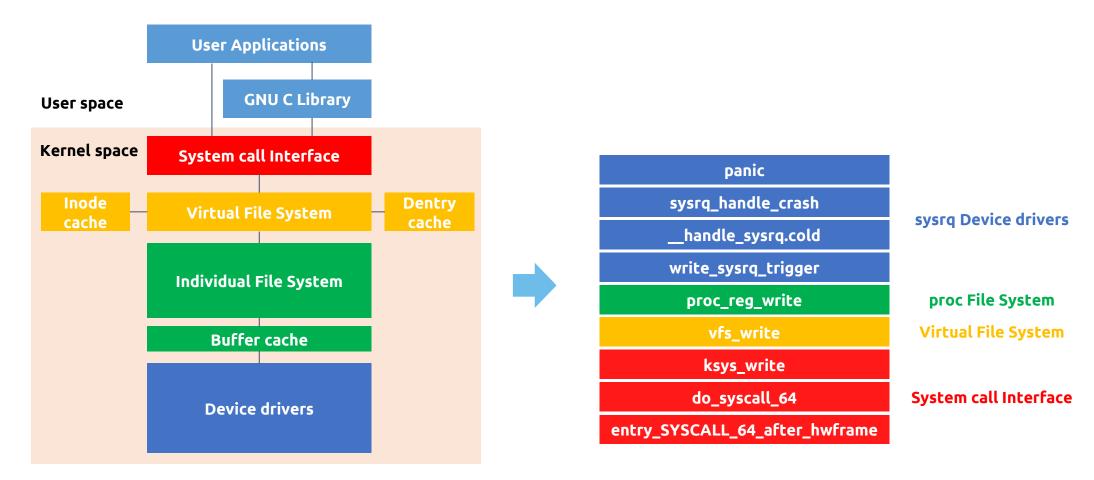


write "c" on /proc/sysrq-trigger





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: sysrg 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 sysra trigger (drivers/tty/sysra.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 ]---
```

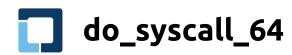
Let's start using based on 6.2.0-rc2!

entry_SYSCALL_64_after_hwframe

```
106 SYM_INNER_LABEL(entry_SYSCALL_64_after_hwfram
                                                       SYM_L_GLOBAL)
                                   /* pt_regs->orig_ax */
       pushq %rax
107
108
       PUSH AND CLEAR REGS rax=$-ENOSYS
109
110
       /* IRQs are off. */
111
112
       movq %rsp, %rdi
       /* Sign extend the lower 32bit as syscall numbers are treated as int */
113
       movslq %eax, %rsi
114
115
       /* clobbers %rax, make sure it is after saving the syscall nr */
116
117
       IBRS ENTER
118
       UNTRAIN_RET
119
       call do_syscall_64
                              /* returns with IRQs disabled */
120
121
```

arch/x86/entry/entry_64.S

panic	
sysrq_handle_crash	sysrq Device drivers
handle_sysrq.cold	
write_sysrq_trigger	
proc_reg_write	proc File System
vfs_write	Virtual File System
ksys_write	
do_syscall_64	System call Interface
entry_SYSCALL_64_after_hwframe	



arch/x86/entry/common.c

panic	
sysrq_handle_crash	sysrq Device drivers
handle_sysrq.cold	
write_sysrq_trigger	
proc_reg_write	proc File System
vfs_write	Virtual File System
ksys_write	
do_syscall_64	System call Interface
entry_SYSCALL_64_after_hwframe	

do_syscall_64

```
73 __visible noinstr void do_syscall_64(struct pt_regs *regs, int nr)
74 {
      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. */
81
          regs->ax = __x64_sys_ni_syscall(regs);
82
83
40 static __always_inline bool do_syscall_x64(struct pt_regs *regs, int nr)
41 {
... SNIP ...
      unsigned int unr = nr;
      if (likely(unr < NR_syscalls)) {
          unr = array_index_nospec(unr, NR_syscalls);
          regs->ax = sys_call_table[unr](regs);
51
          return true;
52
      return false;
54}
```

arch/x86/entry/common.c

panic	
sysrq_handle_crash	sysrq Device drivers
handle_sysrq.cold	
write_sysrq_trigger	
proc_reg_write	proc File System
vfs_write	Virtual File System
ksys_write	
do_syscall_64	System call Interface
entry_SYSCALL_64_after_hwframe	

do_syscall_64

```
40 static __always_inline bool do_syscall_x64(struct pt_regs *regs, int nr)
41 {
... SNIP ...
50 regs->ax = sys_call_table[unr](regs);
... SNIP ...
```

arch/x86/entry/common.c

```
16 asmlinkage const sys_call_ptr_t sys_call_table[] = {
17 #include <asm/syscalls_64.h>
18 };
```

arch/x86/entry/syscall_64.c

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

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

panic	
sysrq_handle_crash	Davies drives
handle_sysrq.cold	sysrq Device drivers
write_sysrq_trigger	
proc_reg_write	proc File System
vfs_write	Virtual File System
ksys_write	
do_syscall_64	System call Interface
entry_SYSCALL_64_after_hwframe	



```
40 static __always_inline bool do_syscall_x64(struct pt_regs *regs, int nr)
41 {
... SNIP ...
50 regs->ax = sys_call_table[unr](regs);
... SNIP ...
```

arch/x86/entry/common.c

```
16 asmlinkage const sys_call_ptr_t sys_call_table[] = {
17 #include <asm/syscalls_64.h>
18 };
```

arch/x86/entry/syscall_64.c

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

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

```
646 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
647 size_t, count)
648 {
649 return ksys_write(fd, buf, count);
650 }
```

sysrq_handle_crash
__handle_sysrq.cold
write_sysrq_trigger

proc_reg_write
vfs_write
ksys_write
do_syscall_64
entry_SYSCALL_64_after_hwframe
sysrq Device drivers

sysrq Device drivers

sysrq Device drivers

System
System
System
Virtual File System
System call Interface

fs/write.c

do_syscall_64

```
73 __visible noinstr void do_syscall_64(struct pt_regs *regs, int nr)
74 {
       add random kstack offset();
       nr = syscall_enter_from_user_mode(regs, nr);
       instrumentation_begin();
79
       if (!do_syscall_x64(regs, nr) && !do_syscall_x32(regs, nr) && nr != -1) {
81
           /* Invalid system call, but still a system call. */
82
           regs->ax = __x64_sys_ni_syscall(regs);
83
40 static always inline bool do_syscall_x64(struct pt_regs *regs, int nr)
41 {
... SNIP ...
       unsigned int unr = nr;
46
47
48
       if (likely(unr < NR_syscalls)) {</pre>
           unr = array_index_nospec(unr, NR_syscalls);
           regs->ax = sys_call_table
51
           return true;
                                               calls ksys_write
52
       return false;
53
54}
```

arch/x86/entry/common.c

```
646 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
647 size_t, count)
648 {
649 return ksys_write(fd, buf, count);
650 }
```

panic

sysrq_handle_crash

__handle_sysrq.cold

write_sysrq_trigger

proc_reg_write

vfs_write

ksys_write

do_syscall_64

entry_SYSCALL_64_after_hwframe

sysrq Device drivers

proc File System
Virtual File System

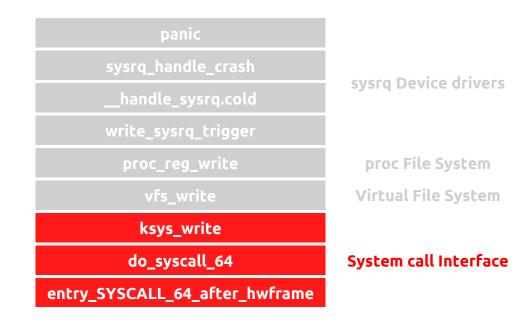
System call Interface

fs/write.c

ksys_write

```
626 ssize_t ksys_write(unsigned int fd, const char __user *buf, size_t count)
627 {
628
        struct fd f = fdget_pos(fd);
629
        ssize tret = -EBADF;
630
        if (f.file) {
631
632
            loff_t pos, *ppos = file_ppos(f.file);
633
            if (ppos) {
634
                pos = *ppos;
635
                ppos = &pos;
636
            ret = vfs_write(f.file, buf, count, ppos);
637
638
            if (ret >= 0 && ppos)
                f.file->f_pos = pos;
639
640
            fdput_pos(f);
641
642
643
        return ret;
644}
```

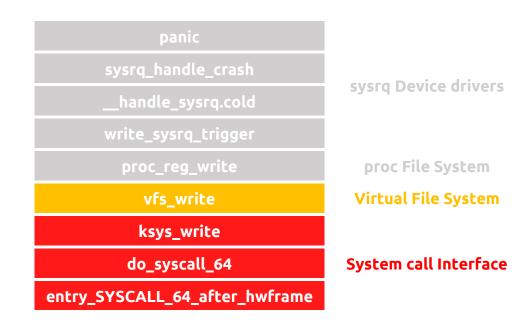
fs/read_write.c



vfs_write

```
564 ssize t vfs_write(struct file *file, const char user *buf, size t count, loff t *pos)
565 {
566
        ssize_t ret;
... SNIP ...
        ret = rw_verify_area(WRITE, file, pos, count);
575
576
        if (ret)
577
            return ret;
578
        if (count > MAX RW COUNT)
579
            count = MAX RW COUNT;
580
        file_start_write(file);
        if (file->f op->write)
581
            ret = file->f_op->write(file, buf, count, pos);
582
        else if (file->f_op->write_iter)
583
584
            ret = new sync write(file, buf, count, pos);
        else
585
586
            ret = -EINVAL;
        if (ret > 0) {
587
            fsnotify_modify(file);
588
589
            add wchar(current, ret);
590
591
        inc syscw(current);
592
        file end write(file);
593
        return ret;
594}
```

fs/read write.c



vfs_write

```
564 ssize t vfs_write(struct file *file, const char user *buf, size t count, loff t *pos)
565 {
566
        ssize_t ret;
 ... SNIP ...
        ret = rw_verify_area(WRITE, file, pos, count);
575
576
        if (ret)
577
            return ret;
578
        if (count > MAX RW COUNT)
                                                                                                                                                   sysra Device drivers
579
            count = MAX RW COUNT;
580
        file start write(file);

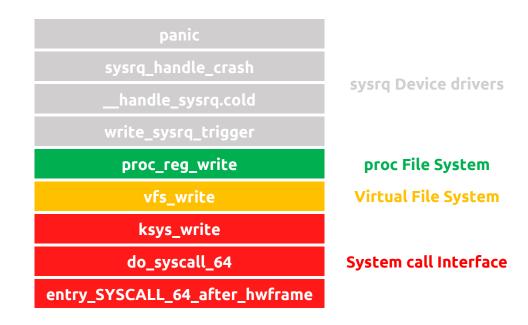
↓ calls proc_reg_write

        if (file->f op->write)
581
            ret = file->f op->write(file, buf, count, pos);
582
        else if (file->f op->write iter)
                                                                                                                                                      proc File System
583
584
            ret = new sync write(file, buf, count, pos);
                                                                                                                     vfs_write
                                                                                                                                                    Virtual File System
        else
585
586
            ret = -EINVAL;
                                                                                                                     ksvs write
587
        if (ret > 0) {
                                                                       578 static const struct file operations proc_reg_file_ops = {
            fsnotify modify(file);
588
                                                                                                                                                        em call Interface
                                                                       579
                                                                               .llseek
                                                                                                   = proc_reg_llseek,
589
            add wchar(current, ret);
                                                                       580
                                                                               .read
                                                                                                   = proc_reg_read,
590
                                                                       581
                                                                               .write
591
        inc syscw(current);
                                                                                                   = proc_reg_write,
                                                                       582
                                                                               .poll
592
        file end write(file);
                                                                                                   = proc_reg_poll,
                                                                               .unlocked_ioctl
                                                                       583
                                                                                                   = proc reg unlocked ioctl,
593
        return ret;
                                                                       584
                                                                               .mmap
                                                                                                   = proc_reg_mmap,
594}
                                                                       585
                                                                               .get_unmapped_area = proc_reg_get_unmapped_area,
                                                                       586
                                                                               .open
                                                                                                   = proc_reg_open,
                                 fs/read write.c
                                                                       587
                                                                               .release
                                                                                                   = proc reg release,
                                                                       588 };
```

proc_reg_write

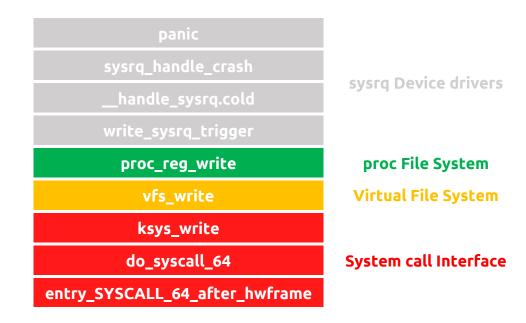
```
344 static ssize_t proc_reg_write(struct file *file, const char __user *buf, size_t count, loff_t *ppos)
345 {
         struct proc_dir_entry *pde = PDE(file_inode(file));
346
         ssize_t rv = -EIO;
347
348
         if (pde_is_permanent(pde)) {
    return pde_write(pde, file, buf, count, ppos);
349
350
         } else if (use_pde(pde)) {
351
             rv = pde_write(pde, file, buf, count, ppos);
352
353
             unuse pde(pde);
354
355
         return rv;
356}
```

fs/proc/inode.c



proc_reg_write

```
344 static ssize_t proc_reg_write(struct file *file, const char __user *buf, size_t count, loff_t *ppos)
345 {
         struct proc_dir_entry *pde = PDE(file_inode(file));
346
347
         ssize trv = -EIO;
348
        if (pde_is_permanent(pde)) {
    return pde_write(pde, file, buf, count, ppos);
349
350
         } else if (use pde(pde)) {
351
352
             rv = pde_write(pde, file, buf, count, ppos);
353
             unuse pde(pde);
354
355
         return rv;
356}
334 static ssize <code>tpde write</code>(struct proc dir entry *pde, struct file *file, const char user *buf,
size tcount, loff t*ppos)
335 {
336
         typeof member(struct proc ops, proc write) write;
337
         write = pde->proc_ops->proc_write;
338
339
         if (write)
             return write(file, buf, count, ppos);
340
341
         return -EIO;
342 }
```



fs/proc/inode.c

proc_reg_write

```
344 static ssize_t proc_reg_write(struct file *file, const char __user *buf, size_t count, loff_t *ppos)
345 {
        struct proc_dir_entry *pde = PDE(file_inode(file));
346
347
        ssize trv = -EIO;
348
        if (pde_is_permanent(pde)) {
    return pde_write(pde, file, buf, count, ppos);
349
350
                                                                                                                                                     sysra Device drivers
        } else if (use pde(pde)) {
351
352
            rv = pde_write(pde, file, buf, count, ppos);
353
            unuse pde(pde);
354
355
        return rv;
                                                                                                                                                       proc File System
                                                                                                                    proc reg write
356}
                                                                                                                       vfs_write
                                                                                                                                                      Virtual File System
334 static ssize t pde write(struct proc dir entry *pde, struct file *file, const char user *buf,
size t count, loff t *ppos)
                                                                                                                      ksys_write
335 {
336
        typeof member(struct proc ops, proc write) write;
                                                                                                                     do_syscall_64
                                                                                                                                                     System call Interface
337
                                                                                                         entry_SYSCALL_64_after_hwframe
338
        write = pde->proc_ops->proc_write; ← calls write_sysrq_trigger
339
        if (write)
            return write(file, buf, count, ppos);
340
        return -EIO;
341
                                                                                   1169 static const struct proc_ops sysrq_trigger_proc_ops = {
342 }
                                                                                                               = write_sysrq_trigger,
                                                                                   1170
                                                                                            .proc write
                                                                                                               = noop llseek,
                                                                                   1171
                                                                                            .proc lseek
                                        fs/proc/inode.c
                                                                                    1172 };
```

write_sysrq_trigger

```
1155 static ssize_t <mark>write_sysrq_trigger</mark>(struct file *file, const char __user *buf,
1156 size_t count, loff_t *ppos)
1157 {
           if (count) {
 1158
 1159
                char c;
1160
 1161
                if (get_user(c, buf))
                     return -EFAULT;
1162
1163
1164
1165
1166
           return count;
1167}
```

drivers/tty/sysrq.c

sysrq_handle_crash
__handle_sysrq.cold
write_sysrq_trigger
proc_reg_write
vfs_write

ksys_write

do_syscall_64
entry_SYSCALL_64_after_hwframe
sysrq Device drivers

sysrq Device drivers

Virtual File System
Virtual File System

__handle_sysrq

```
572 void <u>handle sysrg</u>(int key, bool check mask)
573 {
574
        const struct sysrq_key_op *op_p;
575
        int orig_log_level;
576
        int orig_suppress_printk;
577
        int i;
 ... SNIP ...
        op_p = __sysrq_get_key_op(key);
593
594
        if (op_p) {
595
             * Should we check for enabled operations (/proc/sysrq-trigger
596
597
             * should not) and is the invoked operation enabled?
598
599
            if (!check_mask || sysrq_on_mask(op_p->enable_mask)) {
600
                pr_info("%s\n", op_p->action_msg);
                console_loglevel = orig_log_level;
601
602
                op_p->handler(key);
603
            } else {
604
                pr_info("This sysrq operation is disabled.\n");
605
                console loglevel = orig log level;
606
```

drivers/tty/sysrq.c

sysrq_handle_crash
__handle_sysrq.cold
write_sysrq_trigger

proc_reg_write

vfs_write

ksys_write

do_syscall_64
entry_SYSCALL_64_after_hwframe

sysrq Device drivers

sysrq Device drivers

Virtual File System

Virtual File System

System call Interface

__handle_sysrq

```
572 void <u>handle sysrg</u>(int key, bool check mask)
573 {
574
        const struct sysrq_key_op *op_p;
575
        int orig log level;
        int orig suppress printk;
576
577
        int i;
 ... SNIP ...
        op_p = __sysrq_get_key_op(key);
593
        if (op_p) {
594
                                                                                                                                                   sysra Device drivers
                                                                                                               __handle_sysrq.cold
595
             * Should we check for enabled operations (/proc/sysrq-trigger
596
                                                                                                               write_sysrq_trigger
597
             * should not) and is the invoked operation enabled?
598
                                                                                                                                                      roc File System
                                                                    552 static const struct sysrq_key_op *__sysrq_get_key_op(int key)
            if (!check_mask|| sysrq_on_mask(op_p->enable_mask)) {
599
                                                                    553 {
600
                pr_info("%s\n", op_p->action_msg);
                                                                                                                                                      tual File System
                                                                    554
                                                                            const struct sysrq_key_op *op_p = NULL;
                console loglevel = orig log level;
601
                                                                    555
                                                                            int i;
602
                op_p->handler(key);
                                                                    556
603
            } else {
                                                                    557
                                                                            i = sysrq key table key2index(key);
604
                pr info("This sysrq operation is disabled.\n");
                                                                                                                                                      em call Interface
                                                                    558
                                                                            if (i != -1)
605
                console loglevel = orig log level;
                                                                    559
                                                                                op_p = sysrq_key_table[i];
606
                                                                    560
                                                                    561
                                                                            return op_p;
                                                                   562}
                               drivers/tty/sysrq.c
```

__handle_sysrq

```
572 void <u>handle_sysrq</u>(int key, bool check mask)
573 {
574
        const struct sysrq key op *op p;
575
        int orig log level;
576
        int orig suppress printk;
577
        int i;
... SNIP ...
        op_p = __sysrq_get_key_op(key);
593
                                                                                                               sysrq handle crash
        if (op_p) {
594
                                                                                                                                                   sysra Device drivers
                                                                                                               __handle_sysrq.cold
595
             * Should we check for enabled operations (/proc/sysrq-trigger
596
                                                                                                               write_sysrq_trigger
597
             * should not) and is the invoked operation enabled?
598
                                                                                                                                                       roc File System
                                                                    552 static const struct sysrq key op * sysrq get key op(int key)
599
            if (!check mask || sysrq on mask(op p->enable mask)) {
                                                                    553 {
600
                pr info("%s\n", op p->action msg);
                                                                                                                                                       tual File System
                                                                    554
                                                                            const struct sysrq_key_op *op_p = NULL;
601
                console loglevel = orig log level;
                                                                    555
                                                                            int i;
602
                op p->handler(key);
                                                                    556
603
            } else {
                                                                    557
                                                                            i = sysrq key table key2index(key);
604
                pr info("This sysrq operation is disabled.\n");
                                                                                                                                                       em call Interface
                                                                    558
                                                                            if (i != -1)
605
                console loglevel = orig log level;
                                                                    559
                                                                                op_p = sysrq_key_table[i];
606
                                                                    560
                                                                                                 456 static const struct sysrq_key_op *sysrq_key_table[62] = {
                                                                    561
                                                                            return op_p;
                                                                   562}
                               drivers/tty/sysrq.c
                                                                                                 474
                                                                                                         &sysrq_crash_op,
                                                                                                                                           /* c */
                                                                                                 475
                                                                                                         &sysrq showlocks op,
                                                                                                                                           /* d */
                                                                                                 476
                                                                                                                                           /* e */
                                                                                                         &sysrq term op,
                                                                                                                                           /* f */
                                                                                                 477
                                                                                                         &sysrq_moom_op,
                                                                                                         /* g: May be registered for the kernel debugger */
                                                                                                 478
```

_handle_sysrq

```
572 void <u>handle sysrg</u>(int key, bool check mask)
573 {
574
        const struct sysrq_key_op *op_p;
575
        int orig log level;
        int orig suppress printk;
576
577
        int i;
... SNIP ...
593
        op_p = __sysrq_get_key_op(key);
594
        if (op_p) {
595
             * Should we check for enabled operations (/proc/sysrq-trigger
596
597
             * should not) and is the invoked operation enabled?
598
            if (!check_mask|| sysrq_on_mask(op_p->enable_mask)) {
599
600
                pr_info("%s\n", op_p->action_msg);
601
                console loglevel = orig log level;
602
                op_p->handler(key); ← calls sysrq_handle_crash
603
            } else {
604
                pr_info("This sysrq operation is disabled.\n");
605
                console loglevel = orig log level;
606
                                                                     157 static const struct sysrq_key_op sysrq_crash_op = {
                                                                     158
                                                                            .handler
                                                                     159
                                                                            .help msg
```

sysra Device drivers __handle_sysrq.cold write_sysrq_trigger proc File System proc reg write vfs_write Virtual File System ksys_write do_syscall_64 **System call Interface**

= sysrq_handle_crash,

= SYSRQ ENABLE DUMP,

= "Trigger a crash",

= "crash(c)",

drivers/tty/sysrq.c

160

161

162 };

.action msg

.enable mask

sysrq_handle_crash

```
150 static void sysrq_handle_crash(int key)
151 {
152     /* release the RCU read lock before crashing */
153     rcu_read_unlock();
154
155     panic("sysrq triggered crash\n");
156 }
```

drivers/tty/sysrq.c

sysrq_handle_crash
__handle_sysrq.cold
write_sysrq_trigger

proc_reg_write

vfs_write

ksys_write

do_syscall_64
entry_SYSCALL_64_after_hwframe

sysrq Device drivers

sysrq Device drivers

System

System

System

System call Interface



```
150 static void sysrq_handle_crash(int key)
151 {
152     /* release the RCU read lock before crashing */
153     rcu_read_unlock();
154
155     panic("sysrq triggered crash\n");
156 }
```

drivers/tty/sysrq.c

panic	
sysrq_handle_crash	avesa Davisa deivasa
handle_sysrq.cold	sysrq Device drivers
write_sysrq_trigger	
proc_reg_write	proc File System
vfs_write	Virtual File System
ksys_write	
do_syscall_64	System call Interface
entry_SYSCALL_64_after_hwframe	



echo c > /proc/sysrq-trigger



entry_SYSCALL_64_after_hwframe

do_syscall_64

do_syscall_x64

sys_call_table[unr]()

ksys_write

fs write

file->f_op->write()

proc_reg_write

pde_write

pde->proc_ops->proc_write()

write_sysrq_trigger

__handle_sysrq

op_p->handler();

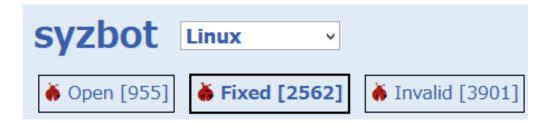
sysrq_handle_crash

Panic X



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

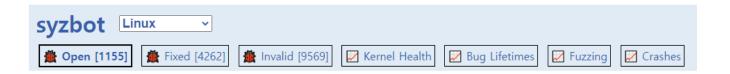


https:///syzkaller.appspot.com/upstream



Information about the bug

when it started happening
when it last happened
how frequently
more crash reports
etc..



open (1071):

<u>Title</u>	Repro	Count	<u>Last</u>	Reported	Last activity
INFO: task hung in ext4 evict ea inode	syz	2	4d02h	2h08m	2h08m
kernel BUG in set extent bit	С	3	4h58m	4h57m	4h57m
memory leak in reiserfs parse options (2)	С	1	4d05h	<u>5h04m</u>	5h04m
WARNING in devlink free		3	15h59m	<u>8h11m</u>	5h48m
KASAN: use-after-free Read in io wq_put and exit		12	15h48m	<u>16h46m</u>	16h40m
WARNING: locking bug in perf event task sched in (2)		11	5h32m	<u>18h58m</u>	5h48m
WARNING: refcount bug in v9fs vfs lookup	С	1	4d21h	<u>21h43m</u>	21h43m
KASAN: stack-out-of-bounds Read in filemap get read batch	syz	1	4d22h	22h27m	22h27m
WARNING in vma complete	syz	2	1d01h	<u>1d07h</u>	1d01h
linux-next boot error: WARNING in drm atomic helper set c		3	5d11h	<u>1d10h</u>	1d10h
kernel BUG in rxrpc put call	С	2	1d21h	<u>1d20h</u>	6h07m
WARNING in bpf xdp adjust tail (4)	С	13	9h39m	<u>1d21h</u>	1d21h
WARNING in shmem evict inode		1	5d22h	<u>1d22h</u>	1d22h
BUG: unable to handle kernel paging request in reiserfs lookup		11	5d07h	<u>2d00h</u>	2d00h
KASAN: use-after-free Read in io wq worker wake	С	1	2d08h	<u>2d08h</u>	1d20h
WARNING in mbind range	C	3	2d01h	<u>2d08h</u>	21h30m
KASAN: slab-out-of-bounds Read in get block (2)		1	2d14h	<u>2d08h</u>	2d08h
general protection fault in ntfs lookup		181	3h04m	<u>2d09h</u>	2d09h
general protection fault in reiserfs get block		2	5d01h	<u>2d09h</u>	2d09h
general protection fault in udf fiiter write fi		1	7d16h	<u>2d09h</u>	2d09h
KASAN: invalid-free in dbUnmount		1	2d20h	<u>2d09h</u>	2d09h



With troubleshooting kernel panic,

I could improve my contribution skills!





Juhee Kang(claudiajkang@gmail.com)