

Linux Crash Dump Analysis

Oliver Yang Jul, 2017

<http://oliveryang.net>

Agenda

- Welcome to panic
- Getting started
- Advanced Studies
- Case Studies
- References

Agenda

- Welcome to panic
- Getting started
- Advanced Studies
- Case Studies
- References

“Computers crash. It's just a fact of life.”

from the “panic” book

Kernel Panic

- An action taken by an operating system upon detecting an internal fatal error from which it cannot safely recover.
 - Kernel printed errors messages via console
 - Freeze the system to dump the memory to dump device
 - Reboot
 - Save vmcore file to filesystem

Causes

- Bugs in kernel, modules or drivers
 - Panic
 - Exception or trap
 - Assertion or BUG_ON
 - Oops
 - If panic_on_oops gets set
- Triggers for system, process hang
 - Watchdogs
 - Sysrq
- Fatal hardware errors...
 - Intel MCE (Machine Check Exceptions)
 - PCIE Uncorrectable Errors
 - Firmware bugs

Debug Information

- Console messages
- System's configurations
 - Hardware
 - Software (kernel, apps)
- Full system logs around the panic time
- Kernel core dump
- Other information dump...
 - MCE banks and registers
 - PCIe TLP packets and registers
 - IPMI SEL logs (BIOS and SMI logs)

STB (Service Tool Bundle) is developed for Solaris and Linux OSes vendors

Kernel Crash Dump

- Memory snapshot while kernel panic
 - Kernel pages only (by default)
 - User pages dump is supported
 - Data on swap device is not included
 - Memory consistence
 - Dump after death
 - Very small partial of memory may not be consistent
 - Depends on dump mechanisms
 - Live dump is always not consistent!
- Kernel dump utilities
 - Kdump (mainline)
 - Disk dump
 - Net dump

Agenda

- Welcome to panic
- Getting started
- Advanced Studies
- Case Studies
- References

Panic – Console Messages

```
[3265705.369809] BUG: unable to handle kernel NULL pointer dereference at 0000000000000011
[3265705.377625] IP: [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.385085] PGD 1c65328067 PUD 2f4818d067 PMD 0
[3265705.391169] Oops: 0000 [#1] SMP
[3265705.395814] Modules linked in: kpatch_s8wspkpx(OE) nbd(OE) tcp_diag inet_diag sch_dsmark act_
[3265705.476305] dca [last unloaded: nbd]
[3265705.480030] CPU: 28 PID: 51681 Comm: test Tainted: G          W OE K----- T 3.10.0-327.
[3265705.492495] Hardware name: Lenovo ThinkServer SD350X/B900G3-10G-N, BIOS A2.16 07/08/2016
[3265705.502088] task: ffff880114fd6480 ti: ffff882f6d8d4000 task.ti: ffff882f6d8d4000
[3265705.511088] RIP: 0010:[<ffffffff811d17f3>] [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.521114] RSP: 0018:ffff882f6d8d7dc8 EFLAGS: 00010286
[3265705.527937] RAX: ffff8808756bb600 RBX: ffff882f6f452000 RCX: ffff8808756bb610
[3265705.536586] RDX: ffff882f6ebaca00 RSI: 0000000000000000 RDI: ffff882f6ebaca20
[3265705.545244] RBP: ffff882f6d8d7e10 R08: ffff882f6d8d4000 R09: ffff88014fb50079
[3265705.553885] R10: 000000000000000a R11: 0000000000000000 R12: 000000001f4eb000
[3265705.562525] R13: 0000000000000000 R14: ffff8808756bb600 R15: 0000000000000001
[3265705.571164] FS: 00007fcbdf0e9700(0000) GS:ffff882fbf180000(0000) knlGS:0000000000000000
[3265705.580741] CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[3265705.587961] CR2: 0000000000000011 CR3: 0000000785a2c000 CR4: 00000000003407e0
[3265705.596561] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[3265705.605180] DR3: 0000000000000000 DR6: 00000000fffe0ff0 DR7: 0000000000000400
[3265705.613758] Stack:
[3265705.617189] 01ff882f6d8d7dd8 ffff882f40ad6000 ffff882f6d8d7e10 ffff882f40ad6000
[3265705.626167] ffff882f6f452000 000000001f4eb000 0000000000000000 ffff882f40ad6000
[3265705.635131] 0000000000000000 ffff882f6d8d7e68 ffffffff811d227f ffff88021c47c380
[3265705.644108] Call Trace:
[3265705.647945] [<ffffffff811d227f>] memcg_stat_show+0x16f/0x2f0
[3265705.655190] [<ffffffff810f5853>] cgroup_seqfile_show+0x73/0x80
[3265705.662503] [<ffffffff81203e47>] ? seq_buf_alloc+0x17/0x40
[3265705.669442] [<ffffffff8120433a>] seq_read+0xfa/0x3a0
[3265705.675875] [<ffffffff811e033c>] vfs_read+0x9c/0x170
[3265705.682273] [<ffffffff811e0e8f>] Sys_read+0x7f/0xe0
[3265705.688593] [<ffffffff816479c9>] system_call_fastpath+0x16/0x1b
[3265705.695923] Code: 4c 8b 30 eb 0f 0f 1f 00 4c 89 ff e8 28 43 f2 ff 84 c0 75 1e 48 8b 33 4c 89
[3265705.719001] RIP [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.726466] RSP <ffff882f6d8d7dc8>
[3265705.731272] CR2: 0000000000000011
```

Header & Register Dumps

```
[3265705.369809] BUG: unable to handle kernel NULL pointer dereference at 0000000000000011
[3265705.377625] IP: [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.385085] PGD 1c65328067 PUD 2f4818d067 PMD 0
[3265705.391169] Oops: 0000 [#1] SMP
[3265705.395814] Modules linked in tcp_diag inet_diag sch_dsmark sch_ingress binfmt_misc..[snipped]..
[3265705.480030] CPU: 28 PID: 51681 Comm: test Tainted: G      W OE K----- T 3.10.0.x86_64 #1
[3265705.492495] Hardware name: Lenovo ThinkServer SD350X/B900G3-10G-N, BIOS A2.16 07/08/2016
[3265705.502088] task: ffff880114fd6480 ti: ffff882f6d8d4000 task.ti: ffff882f6d8d4000
[3265705.511088] RIP: 0010:[<ffffffff811d17f3>] [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.521114] RSP: 0018:ffff882f6d8d7dc8 EFLAGS: 00010286
[3265705.527937] RAX: ffff8808756bb600 RBX: ffff882f6f452000 RCX: ffff8808756bb610
[3265705.536586] RDX: ffff882f6ebaca00 RSI: 0000000000000000 RDI: ffff882f6ebaca20
[3265705.545244] RBP: ffff882f6d8d7e10 R08: ffff882f6d8d4000 R09: ffff88014fb50079
[3265705.553885] R10: 0000000000000000a R11: 0000000000000000 R12: 000000001f4eb000
[3265705.562525] R13: 0000000000000000 R14: ffff8808756bb600 R15: 0000000000000001
[3265705.571164] FS: 00007fcbdf0e9700(0000) GS:ffff882fbf180000(0000) knlGS:0000000000000000
[3265705.580741] CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[3265705.587961] CR2: 0000000000000011 CR3: 0000000785a2c000 CR4: 00000000003407e0
[3265705.596561] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[3265705.605180] DR3: 0000000000000000 DR6: 00000000fffe0ff0 DR7: 0000000000000400
[..... Snipped.....]
```


Stack & Call Trace

[3265705.613758] Stack:

[3265705.617189] 01ff882f6d8d7dd8 ffff882f40ad6000 ffff882f6d8d7e10 ffff882f40ad6000

[3265705.626167] ffff882f6f452000 000000001f4eb000 0000000000000000 ffff882f40ad6000

[3265705.635131] 0000000000000000 ffff882f6d8d7e68 ffffffff811d227f ffff88021c47c380

[3265705.644108] Call Trace:

[3265705.647945] [<ffffffff811d227f>] memcg_stat_show+0x16f/0x2f0

[3265705.655190] [<ffffffff810f5853>] cgroup_seqfile_show+0x73/0x80

[3265705.662503] [<ffffffff81203e47>] ? seq_buf_alloc+0x17/0x40

[3265705.669442] [<ffffffff8120433a>] seq_read+0xfa/0x3a0

[3265705.675875] [<ffffffff811e033c>] vfs_read+0x9c/0x170

[3265705.682273] [<ffffffff811e0e8f>] SyS_read+0x7f/0xe0

[3265705.688593] [<ffffffff816479c9>] system_call_fastpath+0x16/0x1b

[3265705.695923] Code: 4c 8b 30 eb 0f 0f 1f 00 4c 89 ff e8 28 43 f2 ff 84 c0 75 1e 48 8b 33 4c 89 f7 e8 69 43 f2 ff 48 85 c0 49 89 c6 74 69 4c 8b 78 70 <41> f6 47 10 01 74 d6 4d 85 ff 0f 94 c0 4d 85 e>

[3265705.719001] RIP [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0

[3265705.726466] RSP <ffff882f6d8d7dc8>

[3265705.731272] CR2: 0000000000000011

Type

TSC based timestamp and instruction which caused the panic

[3265705.369809] BUG: unable to handle kernel NULL pointer dereference at 0000000000000011
[3265705.377625] IP: [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.385085] PGD 1c65328067 PUD 2f4818d067 PMD 0
[3265705.391169] Oops: 0000 [#1] SMP
[3265705.395814] Modules linked in: tcp_diag inet_diag sch_dsmark sch_ingre _misc..[snipped]..
[3265705.480030] CPU: 0 PID: 51681 Comm: test Tainted: G
[3265705.492495] Hardware name: Lenovo ThinkServer S

x86 exception-specific exception **error code** for this page fault. See x86 implementation:
int __die(const char *str, struct pt_regs *regs, long err);

Panic messages printed by **show_fault_oops** which is handled by CPU exception handler. The **page fault address** is included.

[3265705.553885] R10: 0000000000000000a R11: 0000000000000000 R12: 000000001f4eb000

	Length	Name	Description
P	1 bit	Present	When set, the page fault was caused by a page-protection violation. When not set, it was caused by a non-present page.
W	1 bit	Write	When set, the page fault was caused by a page write. When not set, it was caused by a page read.
U	1 bit	User	When set, the page fault was caused while CPL = 3. This does not necessarily mean that the page fault was a privilege violation.
R	1 bit	Reserved write	When set, the page fault was caused by reading a 1 in a reserved field.
I	1 bit	Instruction Fetch	When set, the page fault was caused by an instruction fetch.

CPU PID Comm Taint Version

```
[3265705.369809] BUG: unable to handle kernel NULL pointer dereference at 0000000000000011
[3265705.377625] IP: [<ffffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.385085] PGD 1c65328067 PUD 2f4818d067 PMD 0
[3265705.391169] Oops: 00010286
[3265705.395814] Modules linked in tcp_diag inet_diag sch_dsmark sch_ingress binfmt_misc [snipped]
[3265705.480030] CPU: 28 PID: 51681 Comm: test Tainted: G      W OE K----- T 3.10.0.x86_64 #1
[3265705.492495] Hardware name: Lenovo ThinkServer SD350X/B900G3-10G-N, BIOS A2.16 07/08/2016
[3265705.502088] task: ffff880114f46480 ti: ffff882f648d4000 task.ti: ffff882f648d4000
```

Panic Task

```
/**
 * print_tainted - return a string to represent the kernel taint state
 *
 * 'P' - Proprietary module has been loaded.
 * 'F' - Module has been forcibly loaded.
 * 'S' - SMP with CPUs not designed for SMP.
 * 'R' - User forced a module unload.
 * 'M' - System experienced a machine check exception.
 * 'B' - System has hit bad_page.
 * 'U' - Userspace-defined naughtiness.
 * 'D' - Kernel has oopsed before
 * 'A' - ACPI table overridden.
 * 'W' - Taint on warning.
 * 'C' - modules from drivers/staging are loaded.
 * 'I' - Working around severe firmware bug.
 * 'O' - Out-of-tree module has been loaded.
 * 'E' - Unsigned module has been loaded.
 * 'L' - A soft lockup has previously occurred.
 * 'K' - Kernel has been live patched.
 *
 * The string is overwritten by the next call to print_tainted().
 */
const char *print_tainted(void)
{
```

```
static const struct tnt tnts[] = {
{ Taint_Proprietary_Module, 'P', 'G' },
{ Taint_Forced_Module, 'F', 'I' },
{ Taint_Unsafe_Smp, 'S', 'I' },
{ Taint_Forced_Rmmod, 'R', 'I' },
{ Taint_Machine_Check, 'M', 'I' },
{ Taint_Bad_Page, 'B', 'I' },
{ Taint_User, 'U', 'I' },
{ Taint_Die, 'D', 'I' },
{ Taint_Overridden_Acpi_Table, 'A', 'I' },
{ Taint_Warn, 'W', 'I' },
{ Taint_Crap, 'C', 'I' },
{ Taint_Firmware_Workaround, 'I', 'I' },
{ Taint_Oot_Module, 'O', 'I' },
{ Taint_Unsigned_Module, 'E', 'I' },
{ Taint_Softlockup, 'L', 'I' },
{ Taint_Livepatch, 'K', 'I' },
{ Taint_16, '?', 'I' },
{ Taint_17, '?', 'I' },
{ Taint_18, '?', 'I' },
{ Taint_19, '?', 'I' },
{ Taint_20, '?', 'I' },
{ Taint_21, '?', 'I' },
{ Taint_22, '?', 'I' },
{ Taint_23, '?', 'I' },
{ Taint_24, '?', 'I' },
{ Taint_25, '?', 'I' },
{ Taint_26, '?', 'I' },
{ Taint_27, '?', 'I' },
{ Taint_Hardware_Unsupported, 'H', 'I' },
{ Taint_Tech_Preview, 'T', 'I' },
};
```

HW FW Registers

[3265705.369809] BUG: unable to handle kernel NULL pointer dereference at 0000000000000011
[3265705.377625] IP: [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.385085] PGD 1c65328067 PUD 2f4818d067 PMD 0
[3265705.391169] Oops: 0000 [#1] SMP
[3265705.395814] Modules linked in tcp_diag inet_diag sch_dsmark sch_ingress binfmt_misc..[snipped]..
[3265705.480030] CPU: 28 PID: 51681 Comm: test Tainted: G W OE K T 3.10.0.x86_64 #1
[3265705.492495] Hardware name: Lenovo ThinkServer SD350X/B900G3-10G-N, BIOS A2.16 07/08/2016
[3265705.502088] task: ffff880114fd6480 ti: ffff882f6d8d4000 task.ti: ffff882f6d8d4000
[3265705.511088] RIP: 0010:<ffffffff811d17f3> [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0
[3265705.521114] RSP: 0018:ffff882f6d8d7dc8 EFLAGS: 00010286
[3265705.527937] RAX: ffff8808756bb600 RBX: ffff882f6f452000 RCX: ffff8808756bb610
[3265705.536586] RDX: ffff882f6ebaca00 RSI: 0000000000000000 RDI: ffff882f6ebaca20
[3265705.545244] RBP: ffff882f6d8d7e10 R08: ffff882f6d8d4000 R09: ffff88014fb50079
[3265705.553885] R10: 0000000000000000a R11: 0000000000000000 R12: 000000001f4eb000
[3265705.562525] R13: 0000000000000000 R14: ffff8808756bb600 R15: 0000000000000001
[3265705.571164] FS: 00007fcbdf0e9700(0000) GS:ffff882fbf180000(0000) knlGS:0000000000000000
[3265705.580741] CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[3265705.587961] CR2: 0000000000000011 CR3: 0000000785a2c000 CR4: 00000000003407e0
[3265705.596561] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[3265705.605180] DR3: 0000000000000000 DR6: 00000000fffe0ff0 DR7: 0000000000000400
[..... Snipped.....]

Exception/Trap Frame

Stack & Call Trace

[3265705.613758] Stack:

The top contents of the stack

[3265705.617188] 01ff882f6d8d7dd8 ffff882f40ad6000 ffff882f6d8d7e10 ffff882f40ad6000
[3265705.621167] ffff882f6f452000 000000001f4eb000 0000000000000000 ffff882f40ad6000
[3265705.631131] 0000000000000000 ffff882f6d8d7e68 ffffffff811d227f ffff88021c47c380

[3265705.641108] Call Trace:

[3265705.641945] [<ffffffff811d227f>] memcg_stat_show+0x16f/0x2f0

[3265705.651190] [<ffffffff810f5853>] cgroup_seqfile_show+0x73/0x80

[3265705.661503] [<ffffffff81203e47>] ? seq_buf_alloc+0x17/0x40

[3265705.661442] [<ffffffff8120433a>] seq_read+0xfa/0x3a0

Back trace

[3265705.671875] [<ffffffff811e033c>] vfs_read+0x9c/0x170

Instruction Address

[3265705.681273] [<ffffffff811e0e8f>] SyS_read+0x7f/0xe0

Code: Instructions

[3265705.681593] [<ffffffff816479c9>] system_call_fastpath+0x16/0x1b

[3265705.691923] Code: 4c 8b 30 eb 0f 0f 1f 00 4c 89 ff e8 28 43 f2 ff 84 c0 75 1e 48 8b 33 4c 89 f7 e8 69 43
f2 ff 48 85 c0 49 89 c6 74 69 4c 8b 78 70 <41> f6 47 10 01 74 d6 4d 85 ff 0f 94 c0 4d 85 e>

[3265705.711001] RIP [<ffffffff811d17f3>] mem_cgroup_iter+0x163/0x2b0

[3265705.721466] RSP [<ffff882f6d8d7dc8>] Stack Pointer

[3265705.731272] CR2: 0000000000000011

Please refer to Documentation/oops-tracing.txt

Crash – Post-mortem Debugging

- A tool for Linux crash dump analysis
 - Maintained by David Anderson from Red Hat
 - Understands all core dump formats
 - Kdump, diskdump, xendump etc.
 - Understands key kernel data structures & meta data
 - Tasks, page table, Slab, files, page cache, devices etc.
 - Also working for live system
- Can be extended via...
 - Patch contributions to core commands
 - Crash extension modules loading by extend command
 - Support gdb scripts
 - Python scripts
 - Eppic scripts
 - See <https://people.redhat.com/anderson/extensions.html>

Invoking Crash

- Running crash with...
 - vmcore or living system
 - Build out vmlinux and modules debug binaries
 - Or kernel-debuginfo & kernel-debuginfo-common-x86_64
 - The **help <command>** shows the manual and examples

```
crash> help
```

*	files	mach	repeat	timer
alias	foreach	mod	runq	tree
ascii	fuser	mount	search	union
bt	gdb	net	set	vm
bttop	help	p	sig	vtop
dev	ipcs	ps	struct	waitq
dis	irq	pte	swap	whatis
eval	kmem	ptob	sym	wr
exit	list	ptov	sys	q
extend	log	rd	task	

Agenda

- Welcome to panic
- Getting started
- **Advanced Studies**
- Case Studies
- References

Panic Process Status

```
crash> ps -l 245407
[208002354263192] [RU] PID: 245407 TASK: ffff88034dacb280 CPU: 7 COMMAND: "kworker/7:0"
crash> bt 245407
PID: 245407 TASK: ffff88034dacb280 CPU: 7 COMMAND: "kworker/7:0"
#0 [ffff88010542ba38] machine_kexec at ffffffff8105327b
#1 [ffff88010542ba98] crash_kexec at ffffffff810f2172
#2 [ffff88010542bb68] oops_end at ffffffff81642558
#3 [ffff88010542bb90] die at ffffffff8101873b
#4 [ffff88010542bbc0] do_trap at ffffffff81641c10
#5 [ffff88010542bc10] do_invalid_op at ffffffff81015214
#6 [ffff88010542bcc0] invalid_op at ffffffff8164b45e
[exception RIP: cgroup_dput+192]
RIP: ffffffff810f9280 RSP: ffff88010542bd78 RFLAGS: 00010246
RAX: 0000000000000002 RBX: ffff880338657bc0 RCX: dead000000200200
RDX: 0000000000000000 RSI: ffff880060d33cf0 RDI: ffff88010c365000
RBP: ffff88010542bda0 R8: ffff880338657c50 R9: dbff0654702af020
R10: dbff0654702af020 R11: 0000000000000781e R12: ffff882e92168780
R13: ffff880060d33cf0 R14: ffff880338657c18 R15: ffff880060d33cf0
ORIG_RAX: ffffffff810f9280 CS: 0010 SS: 0000
#7 [ffff88010542bda8] dentry_kill at ffffffff811f89b6
#8 [ffff88010542bdd8] dput at ffffffff811f8a7c
#9 [ffff88010542bdf8] cgroup_dput at ffffffff810f740c
#10 [ffff88010542be10] css_dput_fn at ffffffff810f743d
#11 [ffff88010542be20] process_one_work at ffffffff810977db
#12 [ffff88010542be68] worker_thread at ffffffff810985ab
#13 [ffff88010542bec8] kthread at ffffffff8109fd0f
#14 [ffff88010542bf50] ret_from_fork at ffffffff81649c18
```

Scheduling timestamp nanoseconds

Scheduling Status: RUN

Panic Location

Stack

Call trace

Panic Location

- RIP could give panic location
- Find source code location by `dis -l`

```
crash> dis -l cgroup_diput+192
/usr/src/debug/kernel-3.10.0/kernel/cgroup.c: 889
0xffffffff810f9280 <cgroup_diput+192>: ud2
```

```
crash> l *cgroup_diput+192
0xffffffff810f9280 is in cgroup_diput (kernel/cgroup.c:889).
884  {
885      /* is dentry a directory ? if so, kfree() associated cgroup */
886      if (S_ISDIR(inode->i_mode)) {
887          struct cgroup *cgrp = dentry->d_fsdata;
888
889          BUG_ON(!(cgroup_is_removed(cgrp)));
```


Stack Dump

○ bt -f

```
ffff88010542bd78: fffff880338657bc0 fffff882e92168780
ffff88010542bd88: fffff880060d33cf0 fffff880338657c18
ffff88010542bd98: 000000000000001c0 fffff88010542bdd0
ffff88010542bda8: ffffffffff811f89b6
[ffff88010542bda8] dentry_kill at ffffffffff811f89b6
ffff88010542bdb0: fffff880338657bc0 fffff880338657c18
ffff88010542bdc0: fffff882fbf2f44c0 fffff882fbf2f9a00
ffff88010542bdd0: fffff88010542bdf0 ffffffffff811f8a7c
[ffff88010542bdd8] dput at ffffffffff811f8a7c
ffff88010542bde0: fffff882f5743a000 fffff882f01437280
ffff88010542bdf0: fffff88010542be08 ffffffffff810f740c
[ffff88010542bdf8] cgroup_dput at ffffffffff810f740c
ffff88010542be00: fffffc90030aaf020 fffff88010542be18
ffff88010542be10: ffffffffff810f743d
[ffff88010542be10] css_dput_fn at ffffffffff810f743d
```

x64 Stack Frame

Position	Contents	Frame
$8n+16$ (%rbp)	memory argument eightbyte n	Previous
	...	
16 (%rbp)	memory argument eightbyte 0	
8 (%rbp)	return address	Current
0 (%rbp)	previous %rbp value	
-8 (%rbp)	unspecified	
	...	
0 (%rsp)	variable size	
-128 (%rsp)	red zone	

Function Prologue

```
crash> dis cgroup_diput 10
```

```
0xffffffff810f91c0 <cgroup_diput>:    nopl  0x0(%rax,%rax,1) [FTRACE NOP]
```

```
0xffffffff810f91c5 <cgroup_diput+5>:  push  %rbp
```

```
0xffffffff810f91c6 <cgroup_diput+6>:  mov   %rsp,%rbp
```

```
0xffffffff810f91c9 <cgroup_diput+9>:  push  %r15
```

```
0xffffffff810f91cb <cgroup_diput+11>: mov   %rsi,%r15
```

```
0xffffffff810f91ce <cgroup_diput+14>: push  %r14
```

```
0xffffffff810f91d0 <cgroup_diput+16>: push  %r13
```

```
0xffffffff810f91d2 <cgroup_diput+18>: push  %r12
```

```
0xffffffff810f91d4 <cgroup_diput+20>: push  %rbx
```

Function Epilogue

```
crash> dis cgroup_diput+144 8
```

```
0xffffffff810f9250 <cgroup_diput+144>: pop    %rbx
```

```
0xffffffff810f9251 <cgroup_diput+145>: pop    %r12
```

```
0xffffffff810f9253 <cgroup_diput+147>: pop    %r13
```

```
0xffffffff810f9255 <cgroup_diput+149>: pop    %r14
```

```
0xffffffff810f9257 <cgroup_diput+151>: pop    %r15
```

```
0xffffffff810f9259 <cgroup_diput+153>: pop    %rbp
```

```
0xffffffff810f925a <cgroup_diput+154>: retq
```

```
0xffffffff810f925b <cgroup_diput+155>: nopl   0x0(%rax,%rax,1)
```


Arguments and return values

- Find arguments and return values from registers
- Exception Frame saves the context of...
 - System call
 - Panic exceptions
- `bt -e`

```
crash> bt -e
PID: 245407 TASK: ffff88034dacb280 CPU: 7 COMMAND: "kworker/7:0"

[exception RIP: cgroup_diput+192]
RIP: ffffffff810f9280 RSP: ffff88010542bd78 RFLAGS: 00010246
RAX: 0000000000000002 RBX: ffff880338657bc0 RCX: dead000000200200
RDX: 0000000000000000 RSI: ffff880060d33cf0 RDI: ffff88010c365000
RBP: ffff88010542bda0 R8: ffff880338657c50 R9: dbff0654702af020
R10: dbff0654702af020 R11: 000000000000781e R12: ffff882e92168780
R13: ffff880060d33cf0 R14: ffff880338657c18 R15: ffff880060d33cf0
ORIG_RAX: ffffffffffffffff CS: 0010 SS: 0000
```

x64 SystemV ABI

Register	Usage	Preserved across function calls
%rax	temporary register; with variable arguments passes information about the number of vector registers used; 1 st return register	No
%rbx	callee-saved register; optionally used as base pointer	Yes
%rcx	used to pass 4 th integer argument to functions	No
%rdx	used to pass 3 rd argument to functions; 2 nd return register	No
%rsp	stack pointer	Yes
%rbp	callee-saved register; optionally used as frame pointer	Yes
%rsi	used to pass 2 nd argument to functions	No
%rdi	used to pass 1 st argument to functions	No
%r8	used to pass 5 th argument to functions	No
%r9	used to pass 6 th argument to functions	No
%r10	temporary register, used for passing a function's static chain pointer	No
%r11	temporary register	No
%r12-r15	callee-saved registers	Yes

Find Arguments From Stack

○ bt -FF

```

RDX: 0000000000000000 RSI: ffff880060d33cf0 RDI: ffff88010c365000
RBP: ffff88010542bda0 R8: ffff880338657c50 R9: dbff0654702af020
R10: dbff0654702af020 R11: 000000000000781e R12: ffff882e92168780
R13: ffff880060d33cf0 R14: ffff880338657c18 R15: ffff880060d33cf0
ORIG_RAX: ffffffffffffffff CS: 0010 SS: 0000
ffff88010542bcc8: [ffff880060d33cf0:inode_cache] [ffff880338657c18:dentry]
ffff88010542bcd8: [ffff880060d33cf0:inode_cache] [ffff882e92168780:dentry]
ffff88010542bce8: ffff88010542bda0 [ffff880338657bc0:dentry]
ffff88010542bcf8: 000000000000781e dbff0654702af020
ffff88010542bd08: dbff0654702af020 [ffff880338657c50:dentry]
ffff88010542bd18: 0000000000000002 dead000000200200
ffff88010542bd28: 0000000000000000 [ffff880060d33cf0:inode_cache]
ffff88010542bd38: [ffff88010c365000:kmalloc-512] ffffffffffffffff
ffff88010542bd48: cgroup_diput+192 0000000000000010
ffff88010542bd58: 0000000000010246 ffff88010542bd78
ffff88010542bd68: 0000000000000000 ffff882fbf2f4cc0
ffff88010542bd78: [ffff880338657bc0:dentry] [ffff882e92168780:dentry]
ffff88010542bd88: [ffff880060d33cf0:inode_cache] [ffff880338657c18:dentry]
ffff88010542bd98: 00000000000001c0 ffff88010542bdd0
ffff88010542bda8: dentry_kill+326
#7 [ffff88010542bda8] dentry_kill at ffffffff811f89b6
ffff88010542bdb0: [ffff880338657bc0:dentry] [ffff880338657c18:dentry]
ffff88010542bdb0: [ffff880338657bc0:dentry] [ffff880338657c18:dentry]
```

Heap/Kmem Dump

```
crash> kmem -s ffff88010c365000
CACHE          NAME          OBJSIZE  ALLOCATED  TOTAL  SLABS  SSIZE
ffff882fbec03600 kmalloc-512      512      28413     29440    920   16k
  SLAB          MEMORY      NODE  TOTAL  ALLOCATED  FREE
  ffffea000430d900 ffff88010c364000    0    32         1    31
  FREE / [ALLOCATED]
  [ffff88010c365000]
crash> cgroup ffff88010c365000
struct cgroup {
  flags = 2,
  count = {
    counter = 1
  },
  tasks = {
    counter = 173
  },
  id = 27,
  sibling = {
    next = 0xfffff882f4de60c28,
    prev = 0xfffff88018b993218
  },
  children = {
    next = 0xfffff88010c365028,
    prev = 0xfffff88010c365028
  },
  files = {
    next = 0xfffff88195c441340,
    prev = 0xfffff88195b2dc800
  },
  parent = 0xfffff882f4de60c00,
  dentry = 0xfffff880338657bc0,
  name = 0xfffff882f01437c80,
  subsys = {0x0, 0x0, 0x0, 0xfffffc90030aaf000, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
```


Raw Memory Dump

```
crash> rd -s ffff88010c365000 48
ffff88010c365000: 0000000000000002 000000ad00000001
ffff88010c365010: 000000000000001b ffff882f4de60c28
ffff88010c365020: ffff88018b993218 ffff88010c365028
ffff88010c365030: ffff88010c365028 ffff88195c441340
ffff88010c365040: ffff88195b2dc800 ffff882f4de60c00
ffff88010c365050: ffff880338657bc0 ffff882f01437c80
ffff88010c365060: 0000000000000000 0000000000000000
ffff88010c365070: 0000000000000000 ffffc90030aaf000
ffff88010c365080: 0000000000000000 0000000000000000
ffff88010c365090: 0000000000000000 0000000000000000
ffff88010c3650a0: 0000000000000000 0000000000000000
ffff88010c3650b0: 0000000000000000 0000000000000000
ffff88010c3650c0: ffff882f5d50c000 ffff881824a4bcc0
ffff88010c3650d0: ffff881824a4bcc0 ffff88002a8ea4d8
ffff88010c3650e0: ffff880147cd44d8 0000000000000000
ffff88010c3650f0: 0000000000000000 ffff88010c3650f8
ffff88010c365100: ffff88010c3650f8 ffff88010c365108
ffff88010c365110: ffff88010c365108 0000000000000001
ffff88010c365120: ffff88010c365120 ffff88010c365120
ffff88010c365130: 0000000000000000 0000000000000000
ffff88010c365140: 0000000000000000 0000000000000000
ffff88010c365150: 0000000ffffffffffe0 ffff88010c365158
ffff88010c365160: ffff88010c365158 cgroup_free_fn
ffff88010c365170: ffff88010c365170 ffff88010c365170
```

Global Variables Dump

```
crash> p irq_stat | head -10
```

```
PER-CPU DATA TYPE:
```

```
    irq_cpustat_t irq_stat;
```

```
PER-CPU ADDRESSES:
```

```
[0]: ffff882fbf212000
```

```
[1]: ffff882fbf232000
```

```
[2]: ffff882fbf252000
```

```
[3]: ffff882fbf272000
```

```
[4]: ffff882fbf292000
```

```
[5]: ffff882fbf2b2000
```

```
[6]: ffff882fbf2d2000
```

The p command could be used directly

```
crash> px irq_stat:1
```

```
per_cpu(irq_stat, 1) = $3 = {
```

```
    __softirq_pending = 0x0,
```

```
    __nmi_count = 0x8549,
```

```
    apic_timer_irqs = 0x550574f,
```

```
    irq_spurious_count = 0x0,
```

```
    icr_read_retry_count = 0x0,
```

```
    kvm_posted_intr_ipis = 0x0,
```

```
    kvm_posted_intr_wakeup_ipis = 0x0,
```

```
    x86_platform_ipis = 0x0,
```

```
    apic_perf_irqs = 0x8548,
```

```
    apic_irq_work_irqs = 0x30b64,
```

```
    irq_resched_count = 0x43572f9,
```

```
    irq_call_count = 0x7c2693,
```

```
    irq_tlb_count = 0x762c84,
```

```
    irq_thermal_count = 0x0,
```

```
    irq_threshold_count = 0x0
```

The px means hex dump
The :1 indicate the data instance on CPU 1.

Global Tasks Status

- Check per-CPU runq and on CPU tasks
 - Runq
 - `bt -a`
- All running/uninterruptable/interruptable tasks
 - `foreach RU bt`
 - `foreach UN bt`
 - `foreach IN bt`
- List kernel/user tasks
 - `ps -k`
 - `ps -u`

Files and IO

- Filesystem information
 - mount
- Who open the files
 - `foreach files -R system.journal`
 - `fuser /usr/lib/libkfm.so.2.0.0`
- Process files and page cache
 - `files 1954`
 - `files -c 1954`
- Block devices and its request queues
 - `dev -d`

Networking

- All NIC interfaces and its net devices
 - `net`
 - `net -n 2618`
- Dump arp cache
 - `net -a`
- All sockets
 - `foreach net -s`

Walkers

- Walk a link list

Using list command, here is an example,

```
crash> dentry.d_name 0xffff882f5a1a8180
```

```
crash> dentry -o 0xffff882f5a1a8180 | grep d_sub
```

```
[ffff882f5a1a8220] struct list_head d_subdirs;
```

```
crash > list -H ffff882f5a1a8220 -o 144 -s dentry.d_name
```

- Walk a tree

- tree -t radix -r address_space.page_tree ffff880bd08412a8

- Create customized walkers by writing scripts

Agenda

- Welcome to panic
- Getting started
- Advanced Studies
- Case Studies
- References

Sysrq Crash

- Make sure crash dump (kdump) works
- Trigger a kernel panic by sysrq
 - `echo c > /proc/sysrq-trigger`
- Waiting for reboot and saving the core file
- Using crash to check the core file
- Explain how sysrq crashes the kernel

Read Files From Page Cache

- Using vi to open a text file in system
- Invoking crash against living system
- Walking the file pages from page cache
- Dump the file contents from memory

See my [page cache debug blog](#).

Agenda

- Welcome to panic
- Getting started
- Advanced Studies
- Case Studies
- References

References

- [Documentation/admin-guide/bug-hunting.rst](#)
- [Documentation/admin-guide/tainted-kernels.rst](#)
- [Crash White Paper](#)
- [x64 System V ABI](#)
- [Page cache debug support for crash](#)