Linux内核调试的实践

讲解时间：3月8日晚9点

宋宝华 <21cnbao@gmail.com>

微信群直播：
http://mp.weixin.qq.com/s/OYMmXnPgOZzfKFeI-8hLpA

扫描二维码报名

麦当劳喜欢您来，喜欢您再来

扫描关注

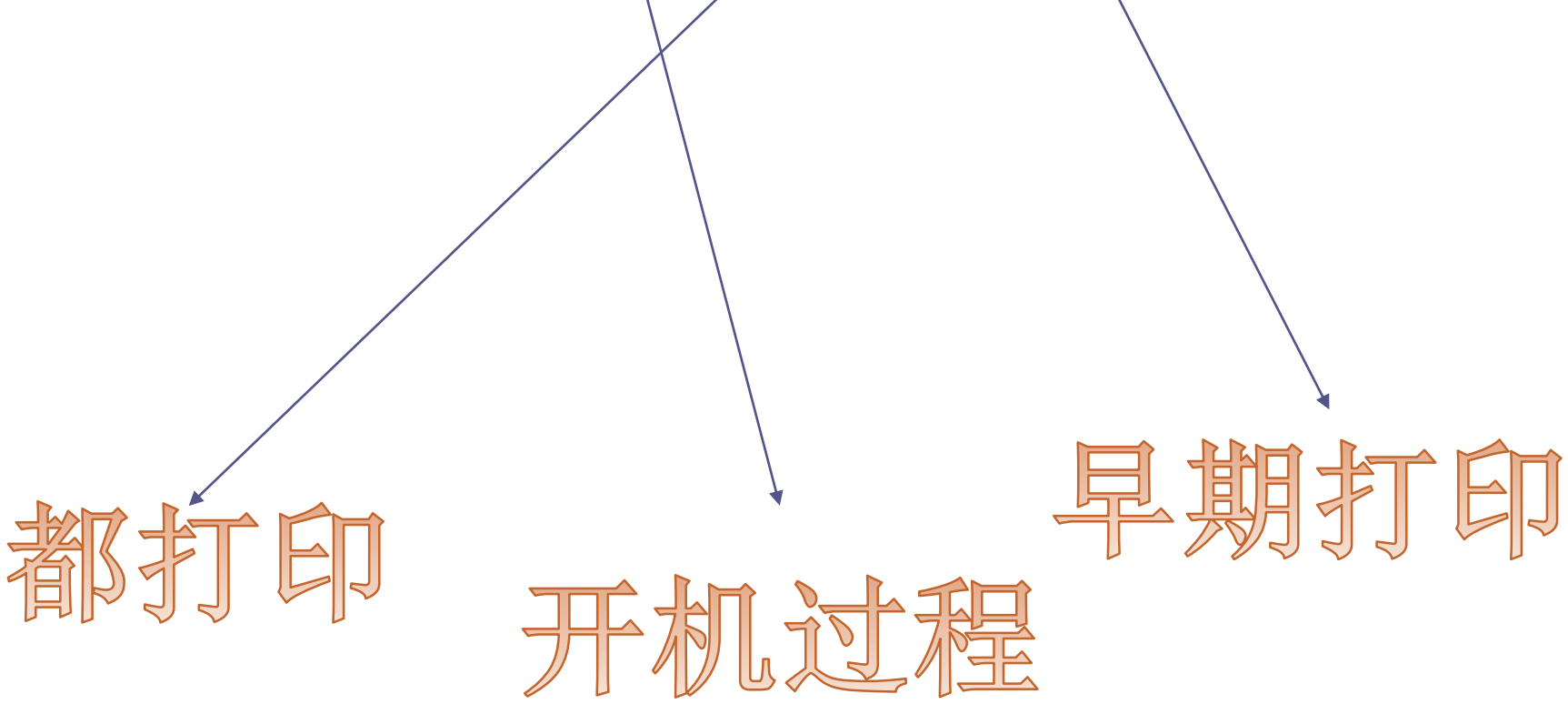
Linuxer

大纲

1.printk解决95%以上的问题
2.工程里的printk，dev_xxx和pr_xxx的正确使用
3.早期的打印
4.开机就死机的调试，initcall_debug
5.printk的耗时，哪些情况可以调用printk
6.printk打印级别控制
7.如何看oops和panic，oops和panic的区别；内核反汇编
8.用gdb对内核进行源代码级调试
9.用gdb对内核模块进行源代码级别的调试
10.用qemu在没有电路板的情况下进行内核源代码级别调试
11.内核内存泄露、内存越界等的调试
12.rcu stalled和系统lockup的调试
13.内核里的各种DEBUG选项
14.grabserial抓开机速度

看一段bootargs

root=/dev/mmcblk0p1 rw rootwait loglevel=8 earlyprintk
console=ttyAMA0 initcall_debug



initcall_debug

IC原厂的Linux工程师经常用它解决开机hang死问题

```
[    0.200519] calling  cpu_suspend_alloc_sp+0x0/0x98 @ 1
[    0.201270] initcall cpu_suspend_alloc_sp+0x0/0x98 returned 0 after 9765 usec
s
[    0.201719] calling  init_static_idmap+0x0/0x118 @ 1
[    0.202101] Setting up static identity map for 0x604ab8c0 - 0x604ab918
[    0.202620] initcall init_static_idmap+0x0/0x118 returned 0 after 0 usecs
[    0.202939] calling  dcscb_init+0x0/0x104 @ 1
[    0.203783] initcall dcscb_init+0x0/0x104 returned -19 after 0 usecs
[    0.204092] calling  tc2_pm_init+0x0/0x154 @ 1
[    0.204644] initcall tc2_pm_init+0x0/0x154 returned -19 after 0 usecs
[    0.204952] calling  spawn_ksoftirqd+0x0/0x28 @ 1
[    0.209451] initcall spawn_ksoftirqd+0x0/0x28 returned 0 after 0 usecs
[    0.209887] calling  init_workqueues+0x0/0x3c4 @ 1
[    0.217764] initcall init_workqueues+0x0/0x3c4 returned 0 after 9765 usecs
[    0.218137] calling  migration_init+0x0/0x74 @ 1
[    0.218536] initcall migration_init+0x0/0x74 returned 0 after 0 usecs
[    0.218864] calling  check_cpu_stall_init+0x0/0x20 @ 1
[    0.219245] initcall check_cpu_stall_init+0x0/0x20 returned 0 after 0 usecs
[    0.219568] calling  rcu_spawn_gp_kthread+0x0/0x158 @ 1
[    0.221489] initcall rcu_spawn_gp_kthread+0x0/0x158 returned 0 after 9765 use
cs
[    0.221962] calling  cpu_stop_init+0x0/0x94 @ 1
[    0.223790] initcall cpu_stop_init+0x0/0x94 returned 0 after 0 usecs
```

early printk的实现

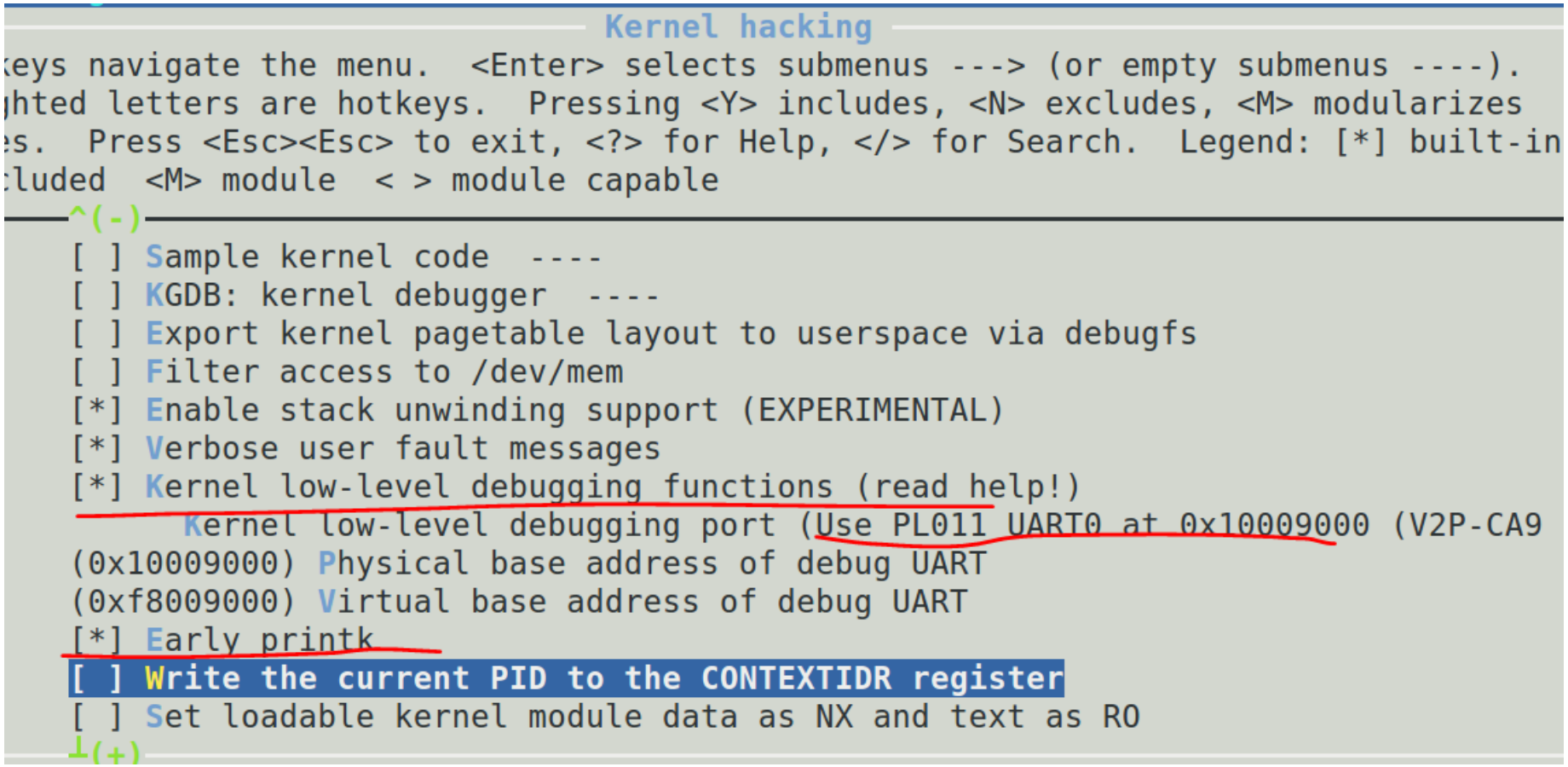
在驱动子系统工作之前，就可以printk

```
baohua@baohua-VirtualBox:~/develop/linux/arch/arm/include/debug$ ls
8250.S        efm32.S     meson.S          s3c24xx.S  tegra.S       zynq.S
asm9260.S     exynos.S    msm.S            s5pv210.S  uncompress.h
at91.S        icedcc.S    netx.S           sa1100.S   ux500.S
bcm63xx.S     imx.S       omap2plus.S      samsung.S  vexpress.S
clps711x.S    imx-uart.h  pl01x.S          sirf.S     vf.S
digicolor.S   ks8695.S    renesas-scif.S   sti.S      vt8500.S
```

往uart的TX寄存器硬塞数据

```
        .macro  addruart, rp, rv, tmp
        ldr     \rp, =CONFIG_DEBUG_UART_PHYS            @ physical
        ldr     \rv, =CONFIG_DEBUG_UART_VIRT            @ virtual
        .endm

        .macro  senduart,rd,rx
        str     \rd, [\rx, #SIRF_LLUART_TXFIFO_DATA]
        .endm

        .macro  busyuart,rd,rx
        .endm

        .macro  waituart,rd,rx
1001:   ldr     \rd, [\rx, #SIRF_LLUART_TXFIFO_STATUS]
        tst     \rd, #SIRF_LLUART_TXFIFO_EMPTY
        beq     1001b
        .endm
```

选 择DEBUG_LL和port



不直接调printk，而是调dev_xxx

设备的名字将作为打印前缀

dev_emerg
dev_warn
dev_info
...

```
acpi/device_pm.c:               dev_info(dev, "System wakeup %s by ACPI\n",
acpi/dock.c:    dev_info(&adev->dev, "ACPI dock station (docks/bays count: %d)\n",
acpi/ioapic.c:          dev_info(&dev->dev, "%s at %pR, GSI %u\n",
acpi/pci_irq.c:                 dev_info(&dev->dev, "PCI IRQ %d -> rerouted to legacy "
acpi/pci_root.c:        dev_info(&root->device->dev, "_OSC: %s [%s]\n", msg, buf);
acpi/pci_root.c:                dev_info(&device->dev, "_OSC failed (%s); disabling ASPM\n",
acpi/pci_root.c:                dev_info(&device->dev, "PCIe port services disabled; not reque
acpi/pci_root.c:                        dev_info(&device->dev,
acpi/pci_root.c:                dev_info(&device->dev, "_OSC failed (%s); disabling ASPM\n",
acpi/pci_root.c:                dev_info(&device->dev, "Disabling ASPM (FADT indicates it is u
acpi/power.c:                   dev_info(&resource->device.dev, "Turning ON\n");
acpi/power.c:                   dev_info(&resource->device.dev, "Turning OFF\n");
acpi/scan.c:                    dev_info(&adev->dev, "Eject disabled\n");
acpi/thermal.c: dev_info(&tz->device->dev, "registered as thermal_zone%d\n",
acpi/video.c:   dev_info(&video->device->dev, "Restoring backlight state\n");
acpi/video.c:   dev_info(&device->dev->dev, "registered as cooling_device%d\n",
ata/acard-ahci.c:                       dev_info(dev, "port %d can do FBS, forcing FBSCP\n",
ata/ahci.c:             dev_info(&pdev->dev, "JMB361 has only one port\n");
ata/ahci.c:             dev_info(&pdev->dev,
ata/ahci.c:             dev_info(&pdev->dev,
ata/ahci.c:     dev_info(host->dev, "applying extra ACPI _GTF filter 0x%x for %s\n",
ata/ahci.c:             dev_info(&pdev->dev,
ata/ahci.c:                     dev_info(&pdev->dev,
ata/ahci.c:             dev_info(&pdev->dev,
```

不直接调printk，而是调pr_xxx

用pr_fmt自定义打印前缀

clk/clk-nomadik.c:#define pr_fmt(fmt) "Nomadik SRC clocks: " fmt
clk/clk-qoriq.c:#define pr_fmt(fmt) KBUILD_MODNAME ": " fmt
clk/spear/clk-aux-synth.c:#define pr_fmt(fmt) "clk-aux-synth: " fmt
clk/spear/clk-frac-synth.c:#define pr_fmt(fmt) "clk-frac-synth: " fmt
clk/spear/clk-gpt-synth.c:#define pr_fmt(fmt) "clk-gpt-synth: " fmt
clk/spear/clk-vco-pll.c:#define pr_

```
huge_memory.c:                  pr_info("raising min_free_kbytes from %d to %lu "
hugetlb.c:              pr_info("HugeTLB registered %s page size, pre-allocated %
hugetlb.c:                      pr_info("Node %d hugepages_total=%u hugepages_fre
hwpoison-inject.c:      pr_info("Injecting memory failure at pfn %#lx\n", pfn);
kmemleak-test.c:        pr_info("kmalloc(32) = %p\n", kmalloc(32, GFP_KERNEL));
kmemleak-test.c:        pr_info("kmalloc(32) = %p\n", kmalloc(32, GFP_KERNEL));
kmemleak-test.c:        pr_info("kmalloc(1024) = %p\n", kmalloc(1024, GFP_KERNEL)
kmemleak-test.c:        pr_info("kmalloc(1024) = %p\n", kmalloc(1024, GFP_KERNEL)
kmemleak-test.c:        pr_info("kmalloc(2048) = %p\n", kmalloc(2048, GFP_KERNEL)
kmemleak-test.c:        pr_info("kmalloc(2048) = %p\n", kmalloc(2048, GFP_KERNEL)
kmemleak-test.c:        pr_info("kmalloc(4096) = %p\n", kmalloc(4096, GFP_KERNEL)
kmemleak-test.c:        pr_info("kmalloc(4096) = %p\n", kmalloc(4096, GFP_KERNEL)
kmemleak-test.c:        pr_info("kmem_cache_alloc(files_cachep) = %p\n",
kmemleak-test.c:        pr_info("kmem_cache_alloc(files_cachep) = %p\n",
kmemleak-test.c:        pr_info("vmalloc(64) = %p\n", vmalloc(64));
kmemleak-test.c:        pr_info("vmalloc(64) = %p\n", vmalloc(64));
kmemleak-test.c:        pr_info("vmalloc(64) = %p\n", vmalloc(64));
kmemleak-test.c:        pr_info("vmalloc(64) = %p\n", vmalloc(64));
kmemleak-test.c:        pr_info("vmalloc(64) = %p\n", vmalloc(64));
kmemleak-test.c:                pr_info("kzalloc(sizeof(*elem)) = %p\n", elem);
kmemleak-test.c:                pr_info("kmalloc(129) = %p\n",
kmemleak.c:             pr_info("%d new suspected memory leaks (see "
kmemleak.c:     pr_info("Automatic memory scanning thread started\n");
kmemleak.c:     pr_info("Automatic memory scanning thread ended\n");
kmemleak.c:             pr_info("Unknown object at 0x%08lx\n", addr);
kmemleak.c:             pr_info("Kmemleak disabled without freeing internal data.
```

不允许这样的打印

FIFO FULL

请问谁的FIFO满了？

Zhang San <san.zhang@x.com>

流芳百世，还是遗臭万年？

```
printk("1\n");
....
printk("2\n");
....
printk("3\n");
```

```
prink("%s %d\n",__func__,
__LINE__);
```

生平见过的最惨烈的printk

```
166 static int __init obsolete_checksetup(char *line)
167 {
168         const struct obs_kernel_param *p;
169         int had_early_param = 0;
170
171         p = __setup_start;
172         printk("172\n");
173         do {
174                 int n = strlen(p->str);
175                 printk("175\n");
176                 if (parameqn(line, p->str, n)) {
177                         if (p->early) {
178                                 /* Already done in parse_early_param?
179                                  * (Needs exact match on param part).
180                                  * Keep iterating, as we can have earl
181                                  * params and __setups of same names 8
182                                 if (line[n] == '\0' || line[n] == '=')
183                                         had_early_param = 1;
184                         } else if (!p->setup_func) {
```

printk其他问题

/proc/sys/kernel/printk运行时候改打印级别

```
#define KERN_EMERG      KERN_SOH "0"
#define KERN_ALERT      KERN_SOH "1"
#define KERN_CRIT       KERN_SOH "2"
#define KERN_ERR        KERN_SOH "3"
#define KERN_WARNING    KERN_SOH "4"
#define KERN_NOTICE     KERN_SOH "5"
#define KERN_INFO       KERN_SOH "6"
#define KERN_DEBUG      KERN_SOH "7"
```

printk中断、软中断、spinlock里面都可以调用

printk的开销很大，尤其是串口打印，会让系统慢很多。
所以产品要在bootargs里面“quiet”

Qemu学习内核源码级调试

qemu-system-arm -s -S -nographic -sd vexpress.img -M vexpress-a9 -m 512M -kernel zImage -dtb vexpress-v2p-ca9.dtb ....

```
baohua@baohua-VirtualBox:~/develop/linux$ arm-linux-gnueabihf-gdb vmlinux
GNU gdb (crosstool-NG linaro-1.13.1-4.8-2013.05 - Linaro GCC 2013.05) 7.6-2013.0
5
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-build_pc-linux-gnu --target=arm-linux-gn
ueabihf".
For bug reporting instructions, please see:
<https://bugs.launchpad.net/gcc-linaro>...
Reading symbols from /home/baohua/develop/linux/vmlinux...done.
(gdb) target remote :1234
Remote debugging using :1234
0x60000000 in ?? ()
(gdb)
```

内核模块源码级调试

```
# modprobe globalmem
[  476.621636] calling  globalmem_init+0x0/0x114
[  476.624723] initcall globalmem_init+0x0/0x114
# cd /sys/module/globalmem/
# ls
coresize     initsize     notes        refcnt
holders      initstate    parameters   sections
# ls -a
.            coresize     initsize     notes
..           holders      initstate    parameters
# cd sections/
# ls -a
.                           .gnu.linkonce.this_mo
..                          .init.text
.ARM.exidx                  .note.gnu.build-id
.ARM.exidx.exit.text        .rodata
.ARM.exidx.init.text        .rodata.str1.4
.alt.smp.init               .strtab
.bss                        .symtab
.data                       .text
.exit.text                  __param
# cat .text
0x7f000000
# cat .data
0x7f000528
#
```

```
(gdb) add-symbol-file drivers/char/globalmem/globalmem.ko 0x7f00000
0 -s .data 0x7f000528
add symbol table from file "drivers/char/globalmem/globalmem.ko" at
        .text_addr = 0x7f000000
        .data_addr = 0x7f000528
(y or n) y
Reading symbols from /home/baohua/develop/linux/drivers/char/global
mem/globalmem.ko...done.
```

Kernel – oops - PC

```
sh-4.2# cat /dev/cma_test
[   33.491284] Unable to handle kernel NULL pointer dereference at virtual address
00000000
[   33.507357] pgd = cc914000
[   33.507379] [00000000] *pgd=0c902831, *pte=00000000, *ppte=00000000
[   33.513769] Internal error: Oops: 817 [#1] PREEMPT
[   33.518538] last sysfs file:
/sys/devices/system/cpu/cpu0/cpufreq/stats/time_in_state
[   33.526350] Modules linked in: vxdkernel pvrsrvkm sirfsoc_gps sirfsocfb cma_test
[   33.533729] CPU: 0    Not tainted  (2.6.38.8-sirf #28)
[   33.538857] PC is at cma_test_read+0x8c/0xec [cma_test]
[   33.544067] LR is at vfs_read+0xb8/0x144
[   33.547966] pc : [<bf06f1cc>]    lr : [<c039d64c>]    psr: a0000013
[   33.547972] sp : cc90ff08  ip : cc90ff30  fp : cc90ff2c
[   33.559421] r10: 00000000  r9 : 00000003  r8 : beafd890
[   33.564631] r7 : cc90ff68  r6 : bf06f4ec  r5 : 00000000  r4 : d3e16460
[   33.571139] r3 : 00000000  r2 : 00000000  r1 : beafd890  r0 : cc8e6e20
[   33.577650] Flags: NzCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment
user
[   33.584768] Control: 10c53c7d  Table: 0c914059  DAC: 00000015
[   33.590495]
[   33.590498] LR: 0xc039d5cc:
```

Kernel – oops -backtrace

```
[   34.152437] Backtrace:
[   34.154882] [<bf06f140>] (cma_test_read+0x0/0xec [cma_test]) from [<c039d64c>]
(vfs_read+0xb8/0x144)
[   34.163982]  r6:beafd890 r5:cc8e6e20 r4:00001000
[   34.168592] [<c039d594>] (vfs_read+0x0/0x144) from [<c039d724>] (sys_read+0x4c/0x108)
[   34.176394]  r8:beafd890 r7:00001000 r6:beafd890 r5:cc8e6e20 r4:000a8c34
[   34.183093] [<c039d6d8>] (sys_read+0x0/0x108) from [<c02dc720>] (ret_fast_syscall+0x0/0x30)
[   34.191416] Code: 03e0002a 0a000012 e59f6058 e3a05000 (e5855000)
[   34.294365] ---[ end trace f0d7620b9f61d90d ]---
```

Kernel – oops - objdump

[33.538857] PC is at cma_test_read+0x8c/0xec [cma_test]

```
00000140 <cma_test_read>:
 140:   e1a0c00d        mov     ip, sp
 144:   e92dd870        push    {r4, r5, r6, fp, ip, lr, pc}
 148:   e24cb004        sub     fp, ip, #4
 14c:   e24dd00c        sub     sp, sp, #12
 150:   e1a0200d        mov     r2, sp
 154:   e3c23d7f        bic     r3, r2, #8128   ; 0x1fc0
 158:   e3c3303f        bic     r3, r3, #63     ; 0x3f
 15c:   e5932004        ldr     r2, [r3, #4]
 160:   e2822001        add     r2, r2, #1
 164:   e5832004        str     r2, [r3, #4]
 168:   e59f30a8        ldr     r3, [pc, #168]  ; 218 <cma_test_read+0xd8>

 1ac:   e3130002        tst     r3, #2
 1b0:   0a000000        beq     1b8 <cma_test_read+0x78>
 1b4:   ebfffffe        bl      0 <preempt_schedule>
 1b8:   e3540000        cmp     r4, #0
 1bc:   03e0002a        mvneq   r0, #42 ; 0x2a
 1c0:   0a000012        beq     210 <cma_test_read+0xd0>
 1c4:   e59f6058        ldr     r6, [pc, #88]   ; 224 <cma_test_read+0xe4>
 1c8:   e3a05000        mov     r5, #0
 1cc:   e5865000        str     r5, [r5]
 1d0:   e5960000        ldr     r0, [r6]
```

反汇编内核里的.o或者.ko

```
baohua@baohua-VirtualBox:~/develop/linux$ arm-linux-gnueabi-objdump -S drivers/char/globalmem/globalmem.o

drivers/char/globalmem/globalmem.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <globalmem_open>:

struct globalmem_dev *globalmem_devp;

static int globalmem_open(struct inode *inode, struct file *filp)
{
        filp->private_data = globalmem_devp;
   0:   e3003000        movw    r3, #0
   4:   e3403000        movt    r3, #0
        return 0;
}
   8:   e3a00000        mov     r0, #0

struct globalmem_dev *globalmem_devp;

static int globalmem_open(struct inode *inode, struct file *filp)
{
```

Oops Vs. Panic

Oops不一定panic
中断上下文的oops会panic
panic_on_oops设置为1，一律panic

阅读：

《宋宝华： Kernel Oops和Panic是一回事吗？》
http://mp.weixin.qq.com/s/cpnE9FNY0xf9n-bzix_abw

Print with timestamp

- grabserial: http://elinux.org/Grabserial

wget http://makelinux.com/emb/*grabserial*

```
[tbird@timdesk data]$ ../grabserial -v -d /dev/ttyUSB1 -e 30 -t -m "Starting kernel.*"
Opening serial port /dev/ttyUSB1
115200:8N1:xonxoff=0:rtcdtc=0
Program will end in 30 seconds
Printing timing information for each line
Matching pattern 'Starting kernel.*' to set base time
Use Control-C to stop...
[0.000001 0.000001]
[0.000433 0.000432]
[0.001908 0.001475] Texas Instruments X-Loader 1.41 (Sep  1 2010 - 13:43:00)
[0.295940 0.294032] mmc read: Invalid size
[0.299905 0.003965] Starting OS Bootloader from MMC/SD1 ...
[0.311140 0.011235]
[0.313113 0.001973]
[0.313168 0.000055] U-Boot 1.1.4-gcebe815a-dirty (Aug 16 2010 - 10:34:46)
[0.317314 0.004146]
[0.317353 0.000039] Load address: 0x80e80000
[0.319459 0.002106] DRAM:  512 MB
[0.321147 0.001688] Flash:  0 kB
[0.360937 0.039790] *** Warning - bad CRC, using default environment
[0.366966 0.006029]
[0.387197 0.020231] In:    serial
```

内核有很多DEBUG选项和功能

- 比如调试suspend：

no_console_suspend

PM_DEBUG

make menuconfig里面能搜索到很多DEBUG

```
Symbol: ATM_ENI_DEBUG [=n]
Type  : boolean
Prompt: Enable extended debugging
  Location:
    -> Device Drivers
(5)   -> Network device support (NETDEVICES [=y])
        -> ATM drivers (ATM_DRIVERS [=n])
          -> Efficient Networks ENI155P (ATM_ENI [=n])
  Defined at drivers/atm/Kconfig:53
  Depends on: ATM_DRIVERS [=n] && NETDEVICES [=y] && ATM [=n] && ATM_ENI [=n]


Symbol: ATM_FORE200E_DEBUG [=]
Type  : integer
Prompt: Debugging level (0-3)
  Location:
    -> Device Drivers
```

有些东西不开DEBUG，真的很难调
-其实是无底洞

在spinlock,中断，软中断sleep?

```
Symbol: DEBUG_ATOMIC_SLEEP [=y]
Type  : boolean
Prompt: Sleep inside atomic section checking
  Location:
    -> Kernel hacking
(1)   -> Lock Debugging (spinlocks, mutexes, etc...)
  Defined at lib/Kconfig.debug:1035
  Depends on: DEBUG_KERNEL [=y]
  Selects: PREEMPT_COUNT [=y]
```

SLUB DEBUG

启动参数：

slub_debug root=/dev/mmcblk0
rootwait

作案现场

```
static ssize_t globalfifo_read(struct file *filp, char __user *buf,
                               size_t count, loff_t *ppos)
{
        int ret;
        struct globalfifo_dev *dev = filp->private_data;
        DECLARE_WAITQUEUE(wait, current);

        char * tbuf = kmalloc(32,GFP_KERNEL);
        if(tbuf)  {
                memset(tbuf,0x00,32);
                kfree(tbuf);
                printk("%s\n","free buf" );
                kfree(tbuf);//重复释放内存
        }
```

```
# cat /dev/globalfifo
[   73.576034] free buf
[   73.579117] ===================================================================
[   73.580048] BUG kmalloc-64 (Not tainted): Object already free
[   73.580495] -------------------------------------------------------------------
[   73.580495]
[   73.581313] Disabling lock debugging due to kernel taint
[   73.583059] INFO: Allocated in globalfifo_read+0x54/0x244 age=0 cpu=1 pid=776
[   73.583731]  globalfifo_read+0x54/0x244
[   73.584130]  __vfs_read+0x18/0x4c
[   73.584459]  vfs_read+0x7c/0x100
[   73.584871]  SyS_read+0x40/0x8c
[   73.585196]  ret_fast_syscall+0x0/0x40
[   73.585638] INFO: Freed in globalfifo_read+0x1c4/0x244 age=2 cpu=1 pid=776
[   73.586186]  __vfs_read+0x18/0x4c
[   73.586526]  vfs_read+0x7c/0x100
[   73.586848]  SyS_read+0x40/0x8c
```

被捕获

kmemleak

内核配置

```
[ ] Extend memmap on extra space for more information
[ ] Debug page memory allocations
[ ] Debug object operations
[ ] SLUB debugging on by default
[ ] Enable SLUB performance statistics
[*] Kernel memory leak detector
(400) Maximum kmemleak early log entries (NEW)
< >   Simple test for the kernel memory leak detector
[*]   Default kmemleak to off
[ ] Stack utilization instrumentation
[ ] Debug VM
[ ] Debug access to
```

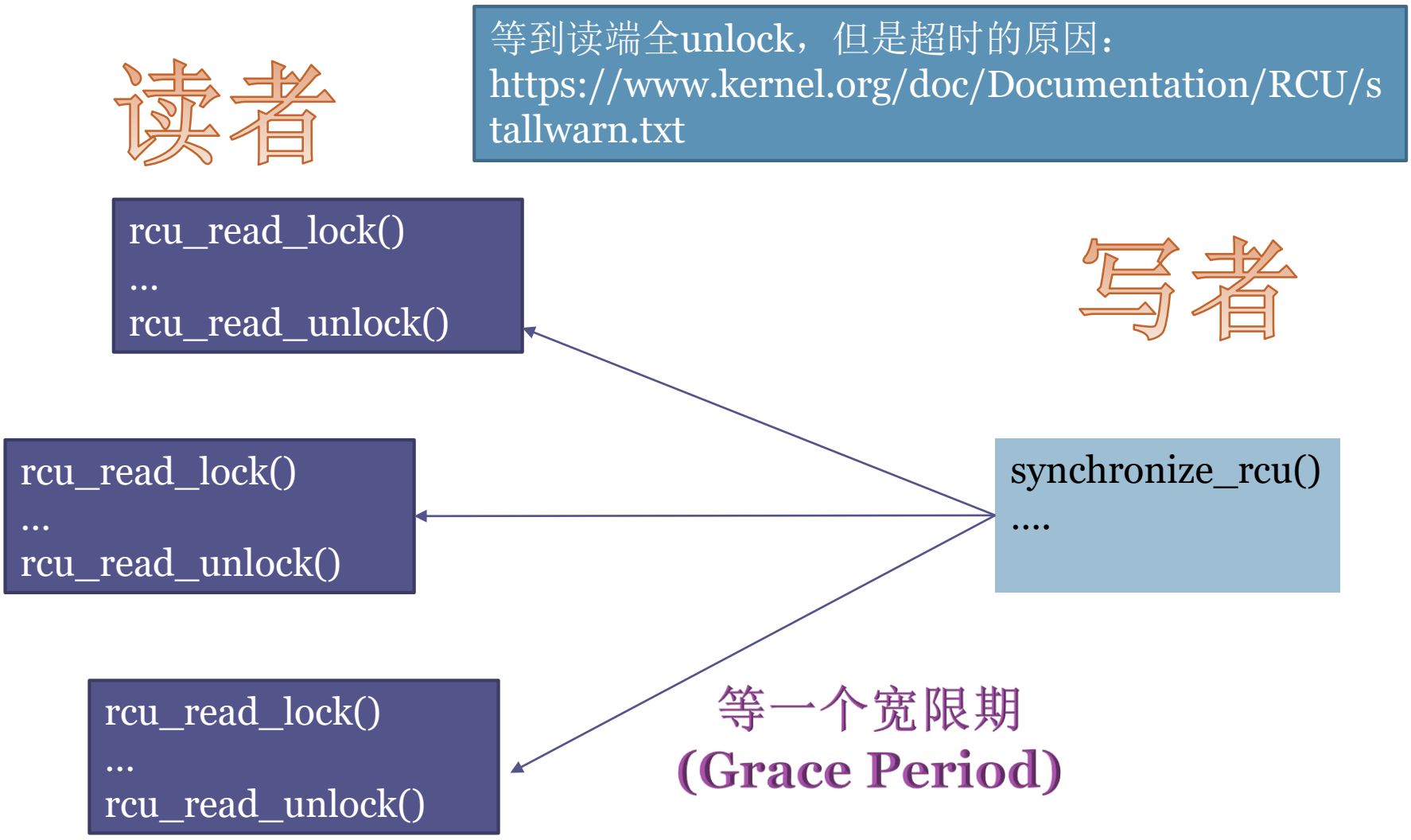
作案现场

```
static int globalfifo_open(struct inode *inode, struct file *filp)
{
        filp->private_data = globalfifo_devp;
        files = get_files_struct(current);


        char * tbuf = kmalloc(32,GFP_KERNEL);
        if(tbuf)  {
                memset(tbuf,0x00,32);
                tbuf = kmalloc(32,GFP_KERNEL); //memory leak
        }

        return 0;
}
```

```
backtrace:
   [<802c5e98>] globalfifo_open+0x54/0x64
   [<800e8ad4>] chrdev_open+0xdc/0x1b0
   [<800e2e6c>] do_dentry_open.isra.12+0xec/0x30c
   [<800efe48>] do_last.isra.38+0x128/0xb8c
   [<800f1ea0>] path_openat+0x7c/0x5cc
   [<800f31a0>] do_filp_open+0x2c/0x80
   [<800e3fa4>] do_sys_open+0x110/0x1cc
   [<8000e720>] ret_fast_syscall+0x0/0x40
   [<ffffffff>] 0xffffffff
unreferenced object 0x9e7f2a00 (size 64):
 comm "cat", pid 780, jiffies 4294947210 (age 13.370s)
 hex dump (first 32 bytes):
   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
 backtrace:
   [<802c5e7c>] globalfifo_open+0x38/0x64
   [<800e8ad4>] chrdev_open+0xdc/0x1b0
   [<800e2e6c>] do_dentry_open.isra.12+0xec/0x30c
```

被抓获！

RCU Stall


读者
等到读端全unlock，但是超时的原因：
https://www.kernel.org/doc/Documentation/RCU/stallwarn.txt
rcu_read_lock()
...
rcu_read_unlock()
写者
rcu_read_lock()
...
rcu_read_unlock()
synchronize_rcu()
....
rcu_read_lock()
...
rcu_read_unlock()
等一个宽限期
(Grace Period)

Lockup detector

- kernel/watchdog.c

✓ NMI中断 + 定时器中断+高优先级RT线程

✓ 用定时器中断，检测高优先级线程有无机会执行->soft lockup

✓ 用NMI，检测定时器中断有无机会执行 -> hard lockup

```
Symbol: HARDLOCKUP_DETECTOR [=n]
Type  : boolean
  Defined at lib/Kconfig.debug:704
  Depends on: LOCKUP_DETECTOR [=y] && !HAVE_NMI_WATCHDOG [=n] && PERF_EVENTS [=y] &&


Symbol: LOCKUP_DETECTOR [=y]
Type  : boolean
Prompt: Detect Hard and Soft Lockups
  Location:
    -> Kernel hacking
(3)   -> Debug Lockups and Hangs
  Defined at lib/Kconfig.debug:680
  Depends on: DEBUG_KERNEL [=y] && !S390
```

Lockup detector案例

代码

```
64 static ssize_t globalmem_read(struct file *filp, char
65                                 loff_t * ppos)
66 {
67         unsigned long p = *ppos;
68         unsigned int count = size;
69         int ret = 0;
70         struct globalmem_dev *dev = filp->private_data
71
72         spinlock_t qlock;
73
74         spin_lock_init(&qlock);
75
76         spin_lock(&qlock);
77         mdelay(30000);
78         spin_unlock(&qlock);
```

Lockup log

```
[  100.291611] NMI watchdog: BUG: soft lockup - CPU#0 stu  for 22s! [cat:716]
[  100.292121] Modules linked in: globalmem
[  100.292924] CPU: 0 PID: 716 Comm: cat Tainted: G           L  4.0.0-rc1+ #47
[  100.293417] Hardware name: ARM-Versatile Express
[  100.293784] task: 9f7cdf00 ti: 9ed32000 task.ti: 9ed32000
[  100.294172] PC is at __loop_delay+0x0/0x10
[  100.294499] LR is at globalmem_read+0x48/0x114 [globalmem]
[  100.294907] pc : [<8023dc38>]    lr : [<7f0001d0>]    psr: 20000013
[  100.294907] sp : 9ed33f28  ip : 8023dc08  fp : 00000000
[  100.295607] r10: 7ee15fa0  r9 : 00001000  r8 : 00001000
[  100.295959] r7 : 9ecda000  r6 : 9ed33f80  r5 : 80659a38  r4 : 00002136
[  100.296375] r3 : 00000000  r2 : 00000e92  r1 : ffffffff  r0 : 0000e856
[  100.296936] Flags: nzCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment user
[  100.297397] Control: 10c5387d  Table: 7ed7806a  DAC: 00000015
[  100.297865] CPU: 0 PID: 716 Comm: cat Tainted: G           L  4.0.0-rc1+ #47
[  100.298301] Hardware name: ARM-Versatile Express
[  100.298667] [<80015790>] (unwind_backtrace) from [<80011a10>] (show_stack+0x10/0x14)
[  100.299162] [<80011a10>] (show_stack) from [<804848e4>] (dump_stack+0x74/0x90)
[  100.299652] [<804848e4>] (dump_stack) from [<8008757c>] (watchdog_timer_fn+0x1a0/0x214)
[  100.300156] [<8008757c>] (watchdog_timer_fn) from [<80065d04>] (__run_hrtimer.isra.19+0x54
[  100.300731] [<80065d04>] (__run_hrtimer.isra.19) from [<80065fa8>] (hrtimer_interrupt+0xd8
[  100.301440] [<80065fa8>] (hrtimer_interrupt) from [<8001459c>] (twd_handler+0x2c/0x40)
[  100.301964] [<8001459c>] (twd_handler) from [<8005a6ac>] (handle_percpu_devid_irq+0x68/0x8
[  100.302494] [<8005a6ac>] (handle_percpu_devid_irq) from [<80056cd8>] (generic_handle_irq+0
[  100.303238] [<80056cd8>] (generic_handle_irq) from [<80056dd4>] (__handle_domain_irq+0x54/
[  100.304247] [<80056dd4>] (__handle_domain_irq) from [<80008670>] (gic_handle_irq+0x20/0x5c
[  100.305211] [<80008670>] (gic_handle_irq) from [<80012500>] (__irq_svc+0x40/0x54)
[  100.305725] Exception stack(0x9ed33ee0 to 0x9ed33f28)
```

更早课程

- 《Linux总线、设备、驱动模型》录播：
http://edu.csdn.net/course/detail/5329

- 深入探究Linux的设备树
http://edu.csdn.net/course/detail/5627

- Linux进程、线程和调度
http://edu.csdn.net/course/detail/5995

- C语言大型软件设计的面向对象
https://edu.csdn.net/course/detail/6496

谢 谢！