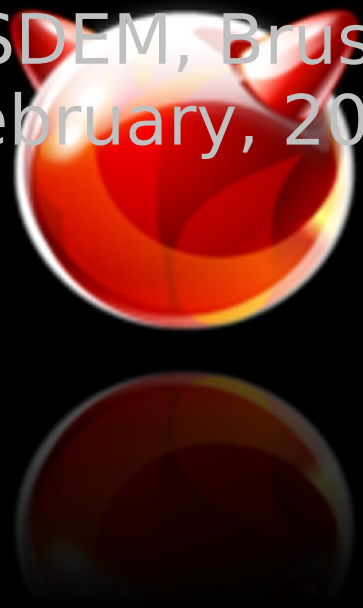
Debugging the FreeBSD kernel for dummies

FOSDEM, Brussels
February, 2010



Shteryana Shopova,
syrinx@FreeBSD.org

Getting started

- Preparing your system for debugging
- Tools in hand
- Reporting kernel problems
- Post-mortem analisys with KGDB
- Common error types leading to panics
- On-line kernel debugging with DDB
- Kernel tracing utilities

Prepare the system

- dumpon(8) and savecore(8)
  - /etc/rc.conf

```
[syrinx@tweety]/(76)% cat /etc/rc.conf | grep dump
dumpdev="/dev/ad4s4b"
dumpdir="/var/crash/"
```

- Remote systems

```
options DDB_UNATTENDED
```

- Virtual machines - Pros and cons
  - Qemu
  - VirtualBox

Build a kernel for debugging

- Kernel configuration

```
makeoptions    DEBUG=-g

#ktrace(1)
options     KTRACE

options     KDB
options     KDB_TRACE
options     DDB
options     GBD

options     INVARIANTS
options     INVARIANT_SUPPORT
options     WITNESS
options     WITNESS_SKIPSPIN
options     LOCK_PROFILING
```

```
#memguard(9) and redzone(9)
options     DEBUG_MEMGUARD
options     DEBUG_REDZONE

#ktr(4)
options     KTR
options     ALQ
options     ALQ_KTR
options     KTR_ENTRIES=4096
options     KTR_COMPILE=KTR_INET
options     KTR_MASK=KTR_INET
options     KTR_CPUMASK=0x3
```

Testing and reporting a panic

- If kernel dies, display driver dies too
- Do not panic the system while fsck (8) is running

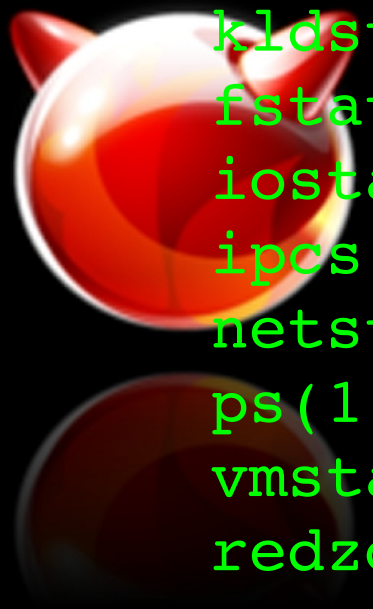
```
[syrinx@tweety]/(78)% cat /etc/rc.conf | grep fsck
fsck_y_enable="YES"
background_fsck="NO"
```

- Core dumps may contain sensitive information

```
# boot -s
# dumpon /dev/ad4s4b
# mount -a -art ufs
# /command-to-crash-the-kernel
```

Tools for further bug analisys

ddb(8)
kgdb(1)
ktr(4)
ktr(9)
ktrdump(8)
ktrace(1)
kdump(1)
dmesg(8)
objdump(1)
size(1)
savecore(8)
dumpon(8)
textdump(8)
hexdump(1)



kgmon(8)
pmcstat(1)
hwpmc(4)
addr2line(1)
kldstat(8)
fstat(1)
iostat(8)
ipcs(1)
netstat(1)
ps(1)
vmstat(8)
redzone(9)
memguard(9)

Post-mortem analisys with kgdb(1)

- Kernel crashes usually require straighforward approach
- kgdb(1) is an extention of gdb(1) that understands FreeBSD kernel corefiles
- Symbol resolving
- Scripting support
- On-line debugging via /dev/mem
- Remote debugging
  - via serial line
  - via firewire

KGDB - Example

```
[syrinx@tweety]/home/syrinx(92)% sudo kgdb /boot/kernel.old/kernel
/var/crash/vmcore.1
GNU gdb 6.1.1 [FreeBSD]

Copyright 2004 Free Software Foundation, Inc.

Unread portion of the kernel message buffer:
rum0: could not multi read MAC register: USB_ERR_NOT_CONFIGURED


Fatal trap 12: page fault while in kernel mode
cpuid = 1; apic id = 01
fault virtual address   = 0x0
fault code              = supervisor read, page not present
instruction pointer     = 0x20:0xc069bf24
stack pointer           = 0x28:0xe6b40c44
frame pointer           = 0x28:0xe6b40c60
code segment            = base 0x0, limit 0xfffff, type 0x1b
                        = DPL 0, pres 1, def32 1, gran 1
processor eflags        = interrupt enabled, resume, IOPL = 0
current process         = 0 (rum0 taskq)
panic: from debugger
```

Common error types leading to panics

- NULL Pointer Dereference
- Resource leak
- Bad test condition, resulting in code path never executed
- Use before test (NULL/-1)
- Use after free
- Buffer overflow
- Unsafe use of returned value
- Uninitialized value read
- Type and allocation size mismatch

Online kernel debugging with DDB

- ddb(4) – interactive kernel debugger
- Poor symbol resolving compared to kgdb(1)
- Extensible – new commands added at compile time -

  Sources at src/sys/ddb
- Scripting support since FreeBSD 7.1 – ddb(8)
- Enter DDB

```
#sysctl debug.kdb.enter=1
```

- Inspecting locks with ddb

```
db> show lock [lockaddr]

db> show turnstile
[lockaddr]

# mutexes

db> show lockchain

# sx locks, sleepqueues

db> show sleepchain
```

- More info in manual page

ktr(4) tracing facility

```
[syrinx@tweety]/home/syrinx(71)% sysctl debug.ktr
debug.ktr.alq_enable: 0
debug.ktr.alq_file: /tmp/ktr.out
debug.ktr.alq_depth: 1024
debug.ktr.alq_failed: 0
debug.ktr.alq_cnt: 0
debug.ktr.alq_max: 0
debug.ktr.clear: 0
debug.ktr.version: 2
debug.ktr.entries: 1024
debug.ktr.compile: 270540806
debug.ktr.mask: 270540806
debug.ktr.cpumask: 3
```

- KTR buffer accessible in DDB

```
db> show ktr [\v]
```

- read the contents of a ktr(4) file

```
[syrinx@tweety]/(105)% sudo ktrdump -t -o /tmp/ktr.out
```

ktrace(1)

- Kernel process tracing

```
[syrinx@tweety]/home/syrinx(89)% ktrace -a -tcy -f
/home/syrinx/krtace.out ifconfig
```

- kdump(1) displays the kernel trace data

```
[syrinx@tweety]/home/syrinx(91)% kdump -f ~/krtace.out
...
7490 ifconfig CALL  write(0x1,0x28203000,0x38)
7490 ifconfig GIO   fd 1 wrote 56 bytes
    options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
7490 ifconfig RET   write 56/0x38
7490 ifconfig CALL  ioctl(0x3,SIOCGIFMAC,0xbfbfdfac)
7490 ifconfig RET   ioctl -1 errno 22 Invalid argument
7490 ifconfig CALL  ioctl(0x3,SIOCGIFMEDIA,0xbfbfdfa0)
...
```

References

- Debugging kernel problems - http://www.lemis.org/grog/papers/Debug-tutorial/tutorial.p
- Introduction to kernel debugging - http://people.freebsd.org/~jhb/papers/bsdcan/2008/article
- FreeBSD Developers' handbook - http://www.freebsd.org/doc/en/books/developers-handbook
- Debugging and profiling the FreeBSD kernel - http://software.intel.com/sites/oss/pdfs/profiling_debugging
- Debugging Buffer overflows in the FreeBSD kernel - http://software/intel.com/sites/oss/pdfs/debugging_buffer_o
- FreeBSD Manual pages

Thank you!

Questions?