UNIVERSITY OF UTAH

STUDENT COMPUTING LABS

You need to restart your computer. Hold down the Power button for seve

Veuillez
démarr
sur le b
puyez

Sie müs
Einscha
Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。

Kernel Panics!

And other nightmares

by James Reynolds

Why?

- Advantages of knowing how to debug
  - The obvious, preventing future panics
  - Not sure what is going on, go into debug mode
  - Looks good on a resume?
  - Bragging rights for sure

What is the kernel?

- Process traffic cop (stop/go), aka scheduling
- Manages memory (gives out memory)
- Speaks hardware input/output
  - Hard Disk
  - Video Card/Display
  - USB - mouse/keyboard/printer/other peripherals
  - Firewire
  - Expansion cards
  - etc

What is a crash?

- Something unexpected happens
  - Car unexpectedly spins out of control
  - Car unexpectedly collides with another car
  - Car unexpectedly collides with wall
  - Car unexpectedly collides into person
  - Car unexpectedly plunges off a cliff & explodes

What is a crash?

- Something unexpected happens
  - Divide by zero
  - Write to memory that isn't yours
  - Try to write to freed memory
  - Access a variable that doesn't exist
  - Put a K of data into a variable that holds a byte

What is a crash?

- "y = z/x" will crash if x = 0
- Good programs check before being bad
  - Ex: `if ( x != 0 ) then { y = z/x }`
- Really good programs will recover
  - Ex: `if ( x != 0 ) then { y = z/x } else { /*recover*/ }`
  - Many programmers are too busy/lazy/ understaffed/inexperienced/distracted/etc
    - So programs crash

What is a crash?

I remarked to Dennis that easily half the code I was writing in Multics was error recovery code. He said, 'We left all that stuff out. If there's an error, we have this routine called panic, and when it is called, the machine crashes, and you holler down the hall, 'Hey, reboot it.'

Lunch conversation between Tom van Vleck and Dennis Rictchie
http://www.multicians.org/unix.html

When the Kernel panics

- Two main causes of kernel panics
  - Hardware problem
    - Bad USB/Firewire/SCSI/PCI interfaces/cards/devices
    - Bad RAM
    - Bad processor, etc
  - Software problem
    - Bad 3rd party driver
    - Bad 3rd party kernel extension
    - Kernel bug

Debugging anyone can do

- KNOW when they happen (for sys admins)
  - Send /Library/Logs/panic.log to yourself!

```
cd /Library/Logs
if [ -e “panic.log” ]; then
     uuencode panic.log panic.log | mail -s panic_log root
     # or
     cat panic.log | mail -s panic_log root
     rm panic.log
fi
```

  - Set up a core dump server (more later)

Debugging anyone can do

- Find bad hardware (RAM, USB, SCSI/PCI, etc)
  - Play the swap game
  - Remove all extra devices
  - Check that all cables are snug
  - Repair hard disk w/ Disk Utility or DiskWarrior
  - Run Apple's Hardware Test CD & other tools
    - TechTool Pro, Memtest/Rember

Debugging anyone can do

- Make sure /System has correct permissions
  - Run Disk Utilities' "Repair Permissions"
- Find bad kernel extensions/hardware drivers
  - Disable all extras (/Library/StartupItems)
  - Check for version compatibilities
  - Safe boot (hold shift after pressing on button)
  - Reinstall OS if last resort

Debugging for superheros

- Read the panic log
- Use gdb
  - 2 machine debugging
  - Set up a panic dump server
- Build custom kernel

Reading the Panic File

- "Understanding and Debugging Kernel Panics"
  - developer.apple.com/technotes/tn2002/tn2063.html
- Log is at /Library/Logs/panic.log
- Look at Backtrace
  - This is the code (in hex) that caused the error
  - You can't read it without gdb (more later)

Reading the Panic File

- Look at the loaded modules
  - Often this will tell you the culprit
- Finding an extension
  - `com.apple.AppleDiskImageController(110)@0x1b76c000`

    `dependency: com.apple.iokit.IOStorageFamily(1.4)@0x1ae3d000`
  - `cd /System/Library/Extentions`

    `grep -r com.apple.AppleDiskImageController *`
  - Result: `IOHDIXController.kext`
    - Loads disk images? (Googling didn't help too much)

UNIVERSITY OF UTAH
STUDENT COMPUTING LABS

Reading the Panic File

- What more can I do?
  - Later on, when running gdb...
  - ```
    cd /System/Library/Extensions
    kextload -s /tmp -n IOHDIXController.kext
    add-symbol-file /tmp/com.apple.AppleDiskImageController.sym
    enter the hexadecimal load addresses for these modules:
    com.apple.iokit.IOStorageFamily: 0x1ae3d000
    com.apple.AppleDiskImageController: 0x1b76c000
    ```
  - You can now get lines numbers in the backtrace for IOHDIXController (instead of ?'s)

UNIVERSITY OF UTAH
STUDENT COMPUTING LABS
    ```

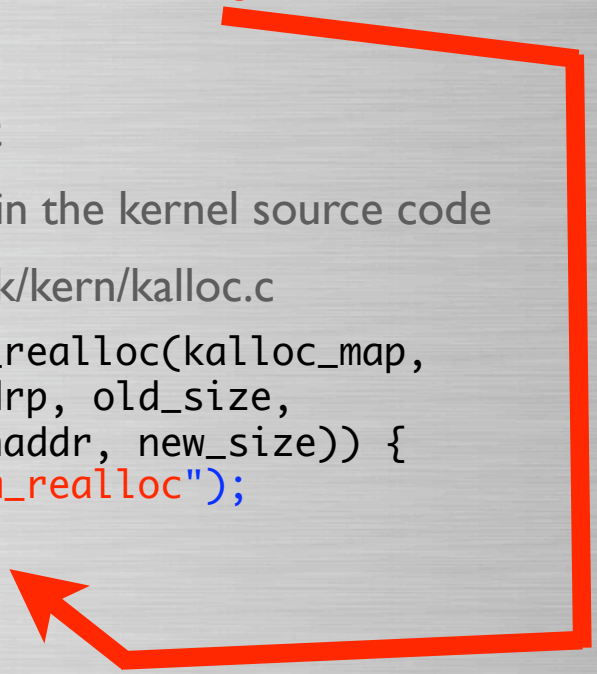
Reading the Panic File

- Reading the first line
  - Two possible messages
    - `panic(cpu 0 caller 0x0025C9A4): message`
      - Anticipated problem occurred!
    - `Unresolved kernel trap(cpu 1): message`
      - CPU or kernel noticed a problem and panicked!
  - The "message" portion tells you quite a bit

Reading the Panic File

- `panic(cpu 0 caller 0x0025C9A4): message`
  - The panic was "on purpose"
  - Copy the message and google it
  - You can find the panic location in the kernel source code
    - Example from xnu-792/osfmk/kern/kalloc.c

```
if (KERN_SUCCESS != kmem_realloc(kalloc_map,
        (vm_offset_t)*addrp, old_size,
        (vm_offset_t *)&naddr, new_size)) {
    panic("krealloc: kmem_realloc");
    naddr = 0;
}
```



UNIVERSITY OF UTAH
STUDENT COMPUTING LABS

Reading the Panic File

`Unresolved kernel trap(cpu 1): message`

- The messages will contain CPU specific info
  - Intel numbers will be different from PowerPC
    - Ex: Intel's 14 = PowerPC's 0x300
- The message wont tell you what led to panic
  - Backtrace does that job
- The message explains what failed
  - Ex: tried to access memory that doesn't exist
  - See docs on the CPU to find out what message means

PowerPC Trap Messages

Unknown
0x100 - System reset
0x200 - Machine check
0x300 - Data access
0x400 - Inst access
0x500 - Ext int
0x600 - Alignment
0x700 - Program
0x800 - Floating point
0x900 - Decrementer
0xA00 - n/a
0xB00 - n/a
0xC00 - System call
0xD00 - Trace
0xE00 - FP assist
0xF00 - Perf mon

PowerPC Trap Messages

0xF20 - VMX
0x1300 - Inst bkpnt
0x1400 - Sys mgmt
0x1600 - Altivec Assist
0x1700 - Thermal
Emulate

0x2000 - Run Mode/Trace
Signal Processor
Preemption
Context Switch
Shutdown
System Failure
INVALID EXCEPTION

Panic Log Examples

```
panic(cpu 0 caller 0x00245B34): BlockAllocateContig:
allocation overflow on "Scratch Disk"
```

Reformat the hard disk for sure!

```
Kernel loadable modules in backtrace (with dependencies):
    com.apple.filesystems.udf(1.4.1)@0x23bf6000
```

Reformating the hard disk stopped this reoccurring panic

Panic Log Examples

```
Kernel loadable modules in backtrace (with dependencies):
    com.apple.driver.AppleUSBEHCI(2.1.5)@0x2a83c000
    dependency: com.apple.iokit.IOUSBFamily(2.1.5)@0x2a7c2000
    dependency: com.apple.iokit.IOPCIFamily(1.4)@0x27d19000
```

USB EHCI is the USB hub and the panic probably occurred when someone unplugged a USB device while it was being mounted (I should report this to Apple)

```
panic(cpu 0 caller 0x000E51BC): bdevvp failed: open
```

No idea. I Googled "bdevvp" and found that it creates a vnode for a block device. So probably a hard disk problem/bug.

Reading the Panic File

- Only so much can be learned from the log
- To get more info, you will have to use gdb!
  - 2 machine debugging
  - Core dump server

2 Machine Debugging

- For reproducible panics
- "Target" the machine that will crash
  - Enable kernel debug mode
- "Host" the computer that you sit at
  - Install Dev Tools, Kernel SDK, xnu source code

UNIVERSITY OF UTAH
STUDENT COMPUTING LABS

Preparing Target

- Enable kernel debug mode
  - It is an Open Firmware setting
    - sudo nvram boot-args=”debug=0x044”
    - Reboot
  - Power button behavior executes NMI
  - Panics wait for connection
  - Other debug settings for different settings
    - See http://developer.apple.com

Preparing Target

- To disable kernel debug mode
  - You want to disable when done!
    - *Anyone* can connect to a panicked machine (with 0x044)
  - sudo nvram boot-args=””
  - Reboot

Preparing Host

- Must be an ADC member!
  - connect.apple.com
- Download latest Dev Tools and install
- Download Kernel Debug SDK
  - developer.apple.com/sdk/
  - Download the OS version you are debugging
  - Mount the disk image

Preparing Host

- Download xnu source code
  - developer.apple.com/darwin/
  - Download the OS version you are debugging
    - Darwin 8.3 = Mac OS X 10.4.3
    - Darwin 8.3's xnu is named xnu-792.6.22
  - Unpack the .tar.gz
  - ```
    sudo mkdir -p /SourceCache/xnu
    sudo ln -s ~/Desktop/xnu-<#> /SourceCache/xnu
    ```
    ```

Reach Out and Touch...

- Target must be panicked
  - To simulate a panic, press the power button
    - Causes Non-Maskable Interrupt (NMI)
      - Target will "freeze"
- On Host
  - ```
    gdb
    target remote-kdp
    attach 10.0.1.1
    ```
    
    replace with target's IP
  - You should now be "in"

UNIVERSITY OF UTAH
STUDENT COMPUTING LABS
    ```

In alien territory

- To leave
  - `detach`
  - NMI machines should return to normal
  - Must restart panicked machines (?)
- To look around
  - ```
    add-symbol-file /Volumes/KernelDebugKit/mach_kernel
    source /Volumes/KernelDebugKit/kgmacros
    bt
    showallstacks
    ```
  - Many more commands... (see developer.apple.com)


    ```

In alien territory

- Email darwin-kernel -at- lists.apple.com
  - Seriously. Those guys are more than willing to tell you want commands you should run and what to look for. You may even be lucky enough to have someone post a patch that will fix the bug so you don't have to wait until the next OS X release.
  - However, you should also post a bug
    - https://bugreport.apple.com

Panic Dump Server

- On targets:
  - `sudo nvram boot-args="debug=0x0d44 _panic_ip=10.0.1.1"`
  - Reboot
- On server:
  - `mkdir /PanicDumps`

    `chmod ugo+w /PanicDumps`

    `pico /etc/xinetd.d/macosxkdump`

Replace with IP of server

Panic Dump Server

```
service macosxkdump
{
    disable     = no
    type        = UNLISTED
    socket_type = dgram
    protocol    = udp
    port        = 1069
    user        = nobody
    groups      = yes
    server      = /usr/libexec/kdumpd
    server_args = /PanicDumps
    wait        = yes
}
```

Panic Dump Server

- On server:
  - `kill -HUP `cat /var/run/xinetd.pid``
  - Cores will be saved in /PanicDumps
    - Names like: core-xnu-792-10.0.1.2-22c3aa51
    - This file will contain a copy of kernel's memory
      - May contain sensitive stuff like passwords

Debugging a Core Dump

- Run these commands

```
gdb -c /PanicDumps/core-xnu-792-10.0.1.2-22c3aa51
```

- Core dump gdb uses different macros
  - http://developer.apple.com/technotes/tn2004/tn2118.html

Building Custom Kernel

- Download DarwinBuild
  - http://opendarwin.org/projects/darwinbuild
  - Build new kernel!
    - `darwinbuild -fetch xnu`
    - Modify source files
    - `darwinbuild xnu`
    - Go to lunch
  - Magically creates custom kernel (universal even)
    - Roots/xnu/xnu-<number>.root~1/mach_kernel

Building Custom Kernel

- Replace your /mach_kernel with new one
  - Make sure permissions are correct!!!
    - root wheel 0644
  - Have spare hard disk ready to boot from in case...
    - You forgot to fix permissions
    - Something else is wrong with it
- Keep your Symbols/xnu/ stuff
  - Use this for debugging future panics

DEMO!!!

- Please fasten your seat belts

UNIVERSITY OF UTAH
STUDENT COMPUTING LABS

Questions & Answers

Any questions or answers?