Kernel Debugging Capture Process

This document explores kernel related problems, how to capture, study and report on them. Generally, these problems can be divided into four broad categories:

1. OOPS
2. Mysterious freezes of hangs
3. Performance problems
4. Data corruption

For each of the above categories of problems, it is important to inform the Red Hat engineer of the exact configuration of the machine displaying the problem. Thus, each bug report should begin with an enumeration of the relevant hardware and software configuration. Listed below is the information required, and the shell command used to obtain the information.

- The exact version and release of the kernel:

cat /proc/version

- The types of IO cards connected to the system:

lspci -vv

- The kernel modules loaded at the time the error was encountered:

lsmod

- The amount of RAM and swap configured in this machine.

cat /proc/meminfo

- The number and type of processors installed in the machine:

cat /proc/cpuinfo

With most kernel problems the machine will likely have been rebooted. The above information is still needed for analysis.

Category 1 Problems: OOPS

An OOPS can indicate either a bug in the kernel code, or a hardware problem. In either case, the text of the OOPS contains very important information. OOPS messages may appear on the machine's console, or they may show up in the machine's system logs. Often, they appear in both places. If you suspect an OOPS, but none is displayed on the console, consult the file */var/log/messages*. The OOPS will appear near the end. If X is running, make sure to check virtual console number one (Ctrl+Alt+F1) to see if the OOPS appears there.

OOPS messages all take the same general form:

- A line explaining the reason the OOPS was generated.
- Additional lines of information which vary by the type of failure.
- The keyword "Oops:"
- A register and stack dump.

OOPS messages are decoded using the **ksymoops(8)** program. Check the manpage for options.

A sample decoded OOPS is shown below.

```
Unable to handle kernel NULL pointer dereference at virtual address
00000018
*pde = 0f992001
Oops: 0000
CPU:    1
EIP:    0010:[]    Not tainted
Using defaults from ksymoops -t elf32-i386 -a i386
EFLAGS: 00010207
eax: 00000000   ebx: c87a1ed0   ecx: c02de5e0   edx: f3de3b00
esi: c87a1eb4   edi: 00000000   ebp: 00000007   esp: c3f5bfa0
ds: 0018   es: 0018   ss: 0018
Process kswapd (pid: 11, stackpage=c3f5b000)
Stack: 00000000 fffffe5d 00000245 00085992 00000001 00000000 000000c0
000000c0
       0008e000 c0136c51 000000c0 00000000 c3f5a000 00000006 c0136ce5
000000c0
       00000000 00010f00 c3ff1fb8 c0105000 c0105866 00000000 c0136c90
c02f5fc0
Call Trace: [] do_try_to_free_pages [kernel] 0x11
[] kswapd [kernel] 0x55
[] stext [kernel] 0x0
[] kernel_thread [kernel] 0x26
[] kswapd [kernel] 0x0
Code: f7 40 18 06 00 00 00 75 f0 8b 40 28 39 d0 75 f0 31 d2 85 d2

EIP; c0136177    <=====
Trace; c0136c51
```

```
Trace; c0136ce5
Trace; c0105000 <_stext+0/0>
Trace; c0105866
Trace; c0136c90
Code;  c0136177
00000000 <_EIP>:
Code;  c0136177    <=====
   0:   f7 40 18 06 00 00 00      testl  $0x6,0x18(%eax)   <=====
Code;  c013617e
   7:   75 f0                     jne    ffffffff9 <_EIP+0xffffffff9>
c0136170
Code;  c0136180
   9:   8b 40 28                  mov    0x28(%eax),%eax
Code;  c0136183
   c:   39 d0                     cmp    %edx,%eax
Code;  c0136185
   e:   75 f0                     jne    0 <_EIP>
Code;  c0136187
  10:   31 d2                     xor    %edx,%edx
Code;  c0136189
  12:   85 d2                     test   %edx,%edx
```

Capturing an OOPS can be a challenge if it does not get recorded in the logs. There are two alternatives. The OOPS can be copied by hand from the console. This is tedious and error prone but is sometimes the only solution.

If the problem is reproducible, however, it can be captured on a remote machine using a serial console setup.

To set up a serial console, attach the serial port of the OOPSing machine to the serial port of another Linux machine using a "null modem" cable. (A regular serial cable won't work.) Add the following to the kernel's boot parameters in the bootloader config file on the OOPSing machine:

```
console=ttyS0,115200 console=tty0
```

and then reboot. (The bootloader config file with either be */etc/lilo.conf* or */boot/grub/grub.conf*, depending on the boot loader chosen usually at installation).

On the second machine, start a serial port client, such as **minicom(1)**, and configure it to run at 115200 baud, parity 8N1. Any output generated by the kernel on the OOPSing machine will be displayed within the serial console client. This information can then be captured and saved in a file on the remote machine.

Another alternative to capturing an OOPS is to use the **netdump** facility, which is available in recently released kernels. If you want to use this facility, you should read the Red Hat Whitepaper on netdump (http://www.redhat.com/support/wpapers/redhat/netdump/) or the documentation on the system in */usr/share/doc/netdump-**, where *netdump-** is the version of netdump you have installed.

Category 2 problems: Mysterious freezes or hangs

These symptoms typically indicate that the kernel is experiencing either *deadlock* or *livelock*. Deadlock occurs when each process in some set of processes is blocked, waiting on a resource held by another process in the set. Livelock occurs when a kernel subsystem is given work to do at a rate greater than it can complete it. These symptoms make the system appear hung, but without producing any OOPS output.

Deadlock problems may manifest themselves as a completely hung system where the user can not change virtual consoles, no keyboard input is echoed on the screen, etc. Alternatively, the system may be responsive, but the user will notice some number of processes stuck in the 'D' state when consulting **ps(1)** or **top(1)**. A good indication of processes in the 'D' state is one with a high load average but low CPU utilization.

In livelock, the system is still active and trying to recover, but no progress is being made by some process or set of processes.

For both of these cases, SysRq should be enabled. SysRq, also know as "The Magic SysRq Key", allows the administrator to send commands directly into the kernel during runtime. To enable SysRq, either do:

echo '1' > /proc/sys/kernel/sysrq

or SysRq can be enabled using **sysctl(8)**.

sysctl -w kernel.sysrq=1

It may be necessary to connect a serial console to the machine to capture the SysRq output.

Once SysRq is enabled, gather the output of Alt+SysRq+T (which dumps the kernel stack of each process) and Alt+SysRq+P (which will show which process is currently executing on each processor).

In addition, it may be useful to reboot with the *non-maskable interrupt watchdog* enabled. To do this, add the line

nmi_watchdog=1

to the bootloader config file and then reboot. The NMI watchdog detects when a processor is hung and automatically generates an OOPS.

Category 3 Problems - Performance problem

This is the vaguest type of problem, and probably one of the most difficult to diagnose. As always, try to gather as much information as possible. Useful information includes:

- What subsystem do you believe is causing the performance problem (network, virtual memory, NFS, etc.)
- Can you tell us specifically why you believe there is a performance problem?
  - Did the problem appear suddenly?
  - Did the problem's appearance correspond to a kernel upgrade?
  - If so, does the problem disappear if you boot into the old kernel?
  - Were there any other changes in configuration that occurred concurrently with the problem's appearance?
- Is there a specific test case which demonstrates the problem?

Category 4 Problems - Data corruption

Because of the enormous amount of testing Red Hat kernels get before they are released, these problems are very rare, even in large installations. However, this is not to say that they are unheard of. In order to diagnose this problem, Red Hat will need to be able to reproduce it. We will require a specific case that will reproduce the problem. Data corruption problems often manifest themselves with OOPS messages, so be certain to check the system logs.

If a test case is not available which reproduces the problem, make sure to submit the exact symptoms of the corruption (i.e. corrupt file contents, destroyed directory, volume won't mount, etc.) It may be a problem of which we are already aware, and perhaps even one for which we have already issued an errata.

Custom kernel support

Custom compiled kernels are not supported. Kernel related errors must be reproduced(able) with a current errata Red Hat kernel.