Kernel Debugging Capture Process

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

1. OOPS
2. Mysterious freezes or 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. Most of the required information can be obtained using the 'sysreport' command (comes in the sysreport rpm package). The following is a list of what should be provided for each of the respective Red Hat Enterprise Linux versions, along with the command used to obtain it:

Red Hat Enterprise Linux 3

- A sysreport from the machine exhibiting the problem:

```
sysreport
```

Red Hat Enterprise Linux 2.1

- A sysreport from the machine exhibiting the problem:

```
sysreport
```

- A list of the types of IO cards connected to the system:

```
lspci -nv
lspci -vv
```

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

Note on custom kernel support

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

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. They often 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:"
- In RHEL3, an lsmod.
- A register and stack dump.

A sample OOPS is shown below.

```
Unable to handle kernel NULL pointer dereference at virtual address 00000f90
printing eip: c020f392
*pde = 35ed8001
*pte = 00000000
Oops: 0000
nfsd lockd sunrpc netconsole e1000 e100 ip_vs_rr ip_vs ipt_mac ipt_REJECT ipt_state
ip_conntrack iptable_filter ip_tables floppy microcode raid5 xor keybdev m
CPU:    1
EIP:    0060:[c020f392]    Not tainted
EFLAGS: 00010202

EIP is at md_update_sb [kernel] 0xb2 (2.4.21-4.ELsmp)
eax: 00000f80   ebx: f0f35680   ecx: 00000000   edx: f0f35680
esi: d3571394   edi: d3571380   ebp: 00000000   esp: e16e9f54
ds: 0068   es: 0068   ss: 0068
Process raid5d (pid: 3873, stackpage=e16e9000)4
Stack: f24aff80 00000003 efd12688 00000064 e16e8000 f6177400 e16e9fb4 efd12688
f89aa762 d3571380 e16e8000 efd12680 e16e9fb4 efd12688 c021255d f6177400
f89abe9b 0219902c e16e8000 00000000 e16e8000 00000000 00000000 00000000
Call Trace:
[f89aa762] raid5d [raid5] 0x142 (0xe16e9f74)
[c021255d] md_thread [kernel] 0x14d (0xe16e9f8c)
[f89abe9b] .rodata.str1.1 [raid5] 0x56 (0xe16e9f94)
[c0212410] md_thread [kernel] 0x0 (0xe16e9fe0)
[c010958d] kernel_thread_helper [kernel] 0x5 (0xe16e9ff0)

Code: f6 40 10 01 75 04 85 c9 74 3a 89 da 8b 1b 39 73 04 75 db 85
```

Capturing an OOPS can be a challenge if it does not get recorded in the logs. There are three options for capturing the data if this happens: hand-copying, serial console, and netdump. If the OOPS occurs suddenly and the possibility for reproduction is unknown, 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 or netdump.

To set up a serial console, attach the serial port of the OOPSing machine to the serial port of another machine using a "null modem" cable (a regular serial cable won't work). Most commonly, a laptop with a null modem cable is used to accomplish this. 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 will either be */etc/lilo.conf* or */boot/grub/grub.conf*, depending on the boot loader running on the machine, which is usually determined during 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. Note: For detailed instructions on the use of minicom, consult minicom's man page ('man minicom').

The third and strongest method for capturing OOPS output is to set up and use the **netdump** facility in your network. The netdump facility may be set up for all servers in your environment to send valuable debugging information, including an OOPS, and send it to a centralized location for easy retrieval. For information on the setup and use of netdump, refer to 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 that a system has many 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
sysctl -p
```

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 the following keystroke combinations, depending on which version of Red Hat Enterprise Linux you are running:

Red Hat Enterprise Linux 3:

- Alt+SysRq+T (dumps the kernel stack of each process)
- Alt+SysRq+W ("walks" the processors and performs the equivalent of an Alt+SysRq+P on each one, which shows what process is currently executing on each processor)
- Alt+SysRq+M (dumps info about what is currently in memory)

Red Hat Enterprise Linux 2.1:

- Alt+SysRq+T (dumps the kernel stack of each process)
- Alt+SysRq+P (shows what process is currently executing on each processor)

  Note: Make sure to issue the Alt+SysRq+P at least twice for each processor on the system.
- Alt+SysRq+M (dumps info about what is currently in memory)

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. **Note!** The NMI watchdog does not work in conjunction with netdump in Red Hat Enterprise Linux 2.1. You may use both at the same time in Red Hat Enterprise Linux 3.

Category 3 Problems - Performance problems

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 go through 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.