Configuring and Analyzing Kernel Crash Dumps

Stefan Seyfried
B1 Systems GmbH
Osterfeldstraße 7
85088 Vohburg
Germany
<seyfried@b1-systems.de>

1 Configuring and Analyzing Kernel Crash Dumps

Did you ever want to investigate that kernel crash on your server but had to reboot quickly to get the system online again? Did you ever encounter a kernel panic which did not get investigated because it left no traces in syslog? A crash dump would probably have helped you.

Get to know the basic steps to configure a Linux system for capturing kernel crash dumps. Even if you are no kernel hacker, that last `dmesg` output of the system can help you locate the problem or even get it fixed by someone else.

2 What are Kernel Crash Dumps?

Kernel crash dumps are a possibility to investigate kernel problems, which can be used even by non-experts to collect all the available information about the problem. This allows a later investigation of the issue by providing the crash dump to your Linux distributor or to a Linux kernel expert. Often it makes it unnecessary to reproduce the problem since all the necessary information is already contained in the crash dump. A crash dump is a complete memory image of the system at the time of the crash, comparable to a core dump of an userspace program.

3 How do Kernel Crash Dumps on Linux Work?

On Linux, the `kdump` facility which in turn uses the system call `kexec` is used to create crash dumps. `kexec` allows to start another Linux sytem – the dump system – out of a running Linux system. In this process, the old Linux system is replaced by the new one, comparable with a quick reboot without boot loader or BIOS. This mechanism prevents the reset of the main memory by the BIOS which would be performed by a regular reboot.

To be able to boot the dump kernel directly upon a critical kernel error, the dump kernel is already loaded in advance using `kexec`. Thus the dump kernel can be started directly without having to load it from the hard drive which might not be accessible anymore without problems. The dump kernel is loaded into a reserved memory area which also is the usable system memory of the dump system.

A so-called "memory hole" is reserved at boot to be available for the dump system in the event of a crash. This is necessary because the dump system must not use the "old" memory in order to not corrupt the image. Possible problems would e.g. be *Direct Memory Access* (DMA) triggered shortly before the crash and still running which could corrupt the memory of the dump kenrel and lead to it also crashing. By denying the "main kernel" access to this memory, such problems are avoided. A size of 128 MB of

reserved memory is usually sufficient. Those 128 MB are no longer available for the productive system. On todays typical x86 system, this should not be a problem.

After the dump kernel has started, the kexec-tools are used to save the old system memory and so create the crash dump. The dump is written as a file to the hard drive or transferred via network to a network share.

The benefit of `kdump` and `kexec` over other crash dump systems is the freshly booted new kernel which provides a stable environment. Other crash dump tools like `lkcd`, `netdump` or `diskdump` have, depending on the kernel problem, not always worked reliably. If the kernel crash was caused by e.g. a network driver problem, there was a high possiblility that sending the crashdump via network was not possible anymore.

4 Configuration Details

To use `kdump`, some prerequisites have to be met.

4.1 Linux Kernel Configuration

Most Linux distributions provide kernels which are "kdump-ready" and have all necessary options set. A description of the configuration depending on the processor architecture is inside the Linux kernel source tree in the file `Documentation/kdump/kdump.txt`. The most important options are `CONFIG_CRASH_DUMP=y` and `CONFIG_KEXEC=y`.

4.2 Boot Parameters

The memory area to be reserved for the dump kernel depends on the processor architecture. It is configured with the kernel parameter `crashkernel=`*`size`*`[@`*`offset`*`]`. *`size`* denotes the size of the memory hole which is later available for the dump system. This size is no longer available for the "normal" running Linux system. *`offset`* sets the physical address in main memory where the memory hole is located and the dump kernel is stored. The *`offset`* is determined by the kernel automatically if it is not given. For x86 and x86_64, `crashkernel=128M@16M` is a usual value.

4.3 Userspace Configuration

To load the dump kernel the program `kexec` from `kexec-tools` is needed. To write the dump out of the dump system, almost all distributions have customized packages which provide integration into the boot process, load the dump kernel and create a special "dump-initrd" if this is necessary. The configuration and the package names are dependent on the distribution.

4.3.1 SUSE Linux / openSUSE

The package containing the kdump helper programs is named `kdump`, the configuration is in the file `/etc/sysconfig/kdump`. The kdump-kernel is loaded with the command `rckdump start` and `chkconfig boot.kdump on` enables automatic loading at each boot. The distribution kernels are kdump-ready. There is a YaST2 module for configuring kdump, it is started with `yast2 kdump`.

4.3.2 Red Hat Linux / Fedora

The kdump-helpers are in the package `crash`, configuration is in `/etc/kdump.conf`. Automatic loading on each boot is achieved with `chkconfig kdump on`. Manual loading of the kdump kernel is done via `service kdump start`. The distribution kernels are kdump-ready. A configuration tool called `system-config-kdump` is available.

4.3.3 Debian

The helper programs are in package `kdump-tools`. The command `kdump-config` allows to check and test the configuration which is done in `/etc/default/kdump-tools`.
Attention: The distribution kernels of Debian are *not* usable as crash kernel, a special kernel needs to be built with `CONFIG_KDUMP=y`. This kernel should then only be used as dump kernel.

4.4 Manual Triggering of a Dumps

For debugging or in the case of abnormal system behaviour triggering a dump manually can be useful. The *Non Maksable Interrupt* (NMI) is a method to do that. Almost all server class hardware has the possibility to manually trigger a NMI, either via the remote management or a button at the machine. This button is often labeled "debug". To have a NMI trigger the crash dump, the Linux kernel needs to be configured. The relevant `sysctl` settingas are `kernel.unknown_nmi_panic` and `kernel.panic_on_unrecovered_nmi`, which should be set to 1. To test the `kdump` configuration, a kernel crash can also be triggered from the running system via the shell:

```
# echo 1 > /proc/sys/kernel/sysrq
# echo c > /proc/sysrq-trigger
```

5 Analyzing the Dump

After a successful crash dump the memory image is at the configured place, often in `/var/crash`. It is possible to simply load such a dump (if it is uncompressed) into the GNU debugger `gdb` and inspect it. However, this method is only advisable if there is lots of knowledge about using `gdb`. Using the `crash` tool is much easier:

```
linux:/var/crash/20111222 # crash System.map-2.6.32.49-0.3-default \
  vmlinux-2.6.32.49-0.3-default.gz vmcore
```

In order to allow `crash` to process the dump correctly, the matching debug information for the kernel needs to be installed. (This examle is of a SUSE system which automatically copies the kernel image and `System.map` into the dump directory. On other systems, those files are typically located in `/boot`.)

5.1 Commands in `crash`

`crash` is mainly a wrapper around `gdb`. Due to this, most of the commands of `gdb` can also be used in `crash`. Additionally, it provides various commands and macros which are specially tailored for Linux crash dumps. A selection of particularly useful commands follows:

5.1.1 help

Maybe the most important command in `crash` is `help`. The help function of `crash` is very extensive and explains every command in detail. Without additional argumes, all `crash` commands are listed. `help` followed by a command provides information about the command. The complete help is displayed with the command `help all`.

5.1.2 log

The `log` displays the log ringbuffer of the kernel. The result is similar to the command `dmesg` in the runnign system. Example (shortened):

```
crash> log
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 2.6.32.49-0.3-default (geeko@buildhost)
(gcc version 4.3.4 [gcc-4_3-branch revision 152973] (SUSE Linux) )
#1 SMP 2011-12-02 11:28:04 +0100
...
[   29.900083] eth0: no IPv6 routers present
[   85.140619] SysRq : Trigger a crash
[   85.140662] BUG: unable to handle kernel NULL pointer dereference
at (null)
...
[   85.141223] RIP  [<ffffffff81282d8d>] sysrq_handle_crash+0xd/0x20
[   85.141248]  RSP <ffff88001d89de80>
[   85.141254] CR2: 0000000000000000
```

Especially the last messages in `log` are often providing strong hints on the reason of the crash. In this example, the crash was triggered intentionally with the key combination *SysRq+C*.

5.1.3 ps

`ps` displays the process list, similar to the corresponding command in the running system. Example:

```
crash> ps
  PID  PPID CPU       TASK        ST  %MEM     VSZ    RSS  COMM
     0    0  0  ffffffff8180c020  RU   0.0       0      0  [swapper]
     1    0  0  ffff88001f1dc040  IN   0.2   10392    792  init
...
  3216    1  0  ffff88001d020280  IN   0.1    4348    756  mingetty
  3217    1  0  ffff88001d8a23c0  IN   0.1    4348    760  mingetty
  3329 3072  0  ffff88001d38a500  IN   0.6   81944   3292  sshd
> 3334 3329  0  ffff88001da74080  RU   0.5   13760   2852  bash
```

A process which was running at the time of the crash is marked with a >.

5.1.4 files

The command `files <PID>` shows information about files opened by a process:

```
crash> files 1795
PID: 1795   TASK: ffff88001cea82c0  CPU: 0   COMMAND: "syslog-ng"
ROOT: /    CWD: /
FD      FILE            DENTRY           INODE       TYPE PATH
0 ffff88001d005b00 ffff88001e84a800 ffff88001d7d6758 CHR  /dev/null
1 ffff88001ce82c00 ffff88001e84a800 ffff88001d7d6758 CHR  /dev/null
2 ffff88001ce82c00 ffff88001e84a800 ffff88001d7d6758 CHR  /dev/null
3 ffff88001d0052c0 ffff88001e9ebd40 ffff88001e8a5708 SOCK
4 ffff88001fb53480 ffff88001e9ebb00 ffff88001e9ea0d8 REG  /var/log/messages
5 ffff88001df6a680 ffff88001e948b00 ffff88001ccdb118 CHR  /dev/tty10
6 ffff88001fa725c0 ffff88001e8a76c0 ffff88001d7d6d98 FIFO /dev/xconsole
7 ffff88001cf55f00 ffff88001eb2b3c0 ffff88001eb28400 REG  /var/log/warn
8 ffff88001df6abc0 ffff88001ebac080 ffff88001ebdfa50 REG  /var/log/mail
```

5.1.5 kmem, vm

`kmem` investigates the memory usage of the kernel, `vm` provides information about the memory usage of individual processes.

5.2 Live Analysis

The `crash` tool can also be used against the currently running linux kernel. This is a good way to explore its capabilities:

```
linux:/boot # crash  vmlinux-2.6.32.49-0.3-default.gz
...

      KERNEL: vmlinux-2.6.32.49-0.3-default.gz
   DEBUGINFO: /usr/lib/debug/boot/vmlinux-2.6.32.49-0.3-default.debug
    DUMPFILE: /dev/mem
        CPUS: 1
        DATE: Fri Dec 30 17:20:45 2011
      UPTIME: 01:37:44
LOAD AVERAGE: 0.03, 0.01, 0.00
       TASKS: 135
    NODENAME: linux
     RELEASE: 2.6.32.49-0.3-default
     VERSION: #1 SMP 2011-12-02 11:28:04 +0100
     MACHINE: x86_64  (1861 Mhz)
      MEMORY: 511.6 MB
         PID: 5342
     COMMAND: "crash"
        TASK: ffff88001da82200  [THREAD_INFO: ffff88001da7c000]
         CPU: 0
       STATE: TASK_RUNNING (ACTIVE)

crash> kmem -i
                 PAGES        TOTAL      PERCENTAGE
 TOTAL MEM   117983     460.9 MB         ----
      FREE     1946       7.6 MB    1% of TOTAL MEM
      USED   116037     453.3 MB   98% of TOTAL MEM
    SHARED    49968     195.2 MB   42% of TOTAL MEM
   BUFFERS     3502      13.7 MB    2% of TOTAL MEM
    CACHED    84339     329.4 MB   71% of TOTAL MEM
      SLAB     7800      30.5 MB    6% of TOTAL MEM

TOTAL SWAP   190762     745.2 MB         ----
 SWAP USED        1         4 KB    0% of TOTAL SWAP
 SWAP FREE   190761     745.2 MB   99% of TOTAL SWAP
crash> files 1
PID: 1      TASK: ffff88001f1c8040  CPU: 0   COMMAND: "init"
ROOT: /    CWD: /
FD       FILE            DENTRY           INODE       TYPE PATH
10 ffff88001dafd9c0 ffff88001e8173c0 ffff88001fad1d98 FIFO /dev/initctl
crash>
```

6 Crash Dumps of Virtual Machines

`crash` can also analyze dumps of virtual machines which are running under the hypervisors *Xen* or *KVM*.

To some extent the hypervisors provide options to automatically create a crash dump in the event of a guest crash. For example with *Xen* the parameter `on_crash='coredump-destroy'` in the VM (domU) configuration file is used to configure that. The biggest benefit over using `kdump` in the guest is the central configuration on the hypervisor without the need to reserve memory inside each guest VM.

Hypervisors also provide the possibility to manually trigger a dump. When using `libvirt` to manage your virtual machines, the command is (independent of the used hypervisor Xen or KVM)

```
# virsh dump domain42 ./domain42.dump
```

This creates a dump of the VM `domain42` into the file `domain42.dump` in the current directory. If Xen is used without `libvirt`, the command is

```
# xm dump-core domain42 ./domain42.dump
```

The possibility of creating a dump of a still running machine can be very helpful in practice, especially if a machine in a production environment which is no longer usable because of yet unknown problems needs a restart. Instead of doing a lenghty problem analysis before the restart the dump allows you to subsquently analyze the problem after the reboot – the downtime is kept as short as possible.

Another noteworthy nice property of manual dumps of virtual machines is, that it is not necessary to configure everything before the fact. If strange behaviour of a VM is noted and if the VM should be restarted because of that, it is possible to decide immediately before the reboot if a dump should be taken.

7 Summary

Crash dumps provide facilities to the sysadmin as well as the kernel hacker to gather useful information in the case of a system crash. Time and effort to set them up is relatively low and is quickly compensated for by the possibility of convenient postmortem analysis.

8 Thanks

This article is based on training material of B1 Systems GmbH which was created by Daniel Gollub and Michael Steinfurth. Further thanks go to Michel Unke, Andreas Steil and Anke Börnig for tireless proofreading and competent help with any TeX problems.