linux > Documentation > kdump > kdump.txt

【linux-2.6.31】kdump --- 基于 kexec 的崩溃转储机制

Translated By: openspace

Date : 2009-12-21

===

术语翻译

dump-capture kernel	转储捕获内核

===

该文档包含概述、配置和安装、以及如何对信息进行分析。

概述
====

每当访问系统内核的内存映像的转储时（比如系统 panics），kdump 利用 kexec 快速启动到一个转储捕获内核。系统内核的内存映像被保留以备重启，并可以通过转储捕获内核来访问。

可以使用普通命令比如 cp 和 scp 来将内存映像复制到本地磁盘的一个转储文件中，或者通过网络复制到远程系统。

当前 kdump 和 kexec 支持 x86、x86_64、ppc64 和 ia64 体系结构。

当系统内核启动时，它会保留少量内存以存放转储捕获内核。这样可以确保系统内核中持续进行的 DMA 操作不会破坏转储捕获内核。命令 **kexec -p** 将转储捕获内核加载到保留的内存空间中。

在 x86 机器上，不管内核加载到哪里，启动时都会用到物理内存的最初 640KB 空间。因此，kexec 在启动进入转储捕获内核之前会将该区域备份。

类似地，在 PPC64 机器上，不管内核加载到哪里，启动时都会用到物理内存的最初 32KB 空间，同时为了支持 64K 页面大小，kexec 会将最初 64KB 空间进行备份。

内核映像的主要信息以 ELF 格式编码，在崩溃之前存放到保留的内存区域中。通过引导参数 **elfcorehdr=**将 ELF header 所处的起始物理地址传送给转储捕获内核。

有了转储捕获内核，可以通过两种方式访问内存映像或者“前一个系统的内存”：

- 通过一个**/dev/oldmem** 设备接口。捕获工具可以以 raw 方式读取设备文件和写入文件。这是内存的一个 raw 转储。分析和捕获工具必须足够“智能”以确定从哪里获取正确的信息
- 通过**/proc/vmcore**。该接口导出的转储采用 ELF 格式，可以使用文件拷贝命令比如 cp 或者 scp 将信息读出来。另外，还可以使用分析工具比如 GDB 和 crash 工具调试转储文件。该方法保证转储的页面顺序没有打乱

配置和安装
==========

安装 kexec-tools

----

1) 以 root 用户身份登录

2) 从下列地址下载 kexec-tools 用户空间包：

http://www.kernel.org/pub/linux/kernel/people/horms/kexec-tools/kexec-tools.tar.gz

这是指向最新版的链接。

可以访问最新的 kexec-tools git 树：

git://git.kernel.org/pub/scm/linux/kernel/git/horms/kexec-tools.git

或者

http://www.kernel.org/git/?p=linux/kernel/git/horms/kexec-tools.git

关于 kexec-tools 的更多信息访问

http://www.kernel.org/pub/linux/kernel/people/horms/kexec-tools/README.html

3) 用 tar 命令解压包

```
tar xvpzf kexec-tools.tar.gz
```

4) 切换到 kexec-tools 目录

```
cd  kexec-tools-VERSION
```

5) 配置./configure

6) 编译 make

7) 安装 make install

构建系统和转储捕获内核

使用 kdump 有两种方式：

1) 构建一个单独的自定义转储捕获内核以捕获内核转储

2) 或者将系统内核本身作为转储捕获内核，这就不需要构建一个单独的转储捕获内核。这只能用于支持可重定位内核的体系结构上；当前 i386、x86_64、ppc64 和 ia64 体系结构支持可重定位内核

构建一个可重定位内核使得不需要构建第二个内核就可以捕获转储。但是可能有时想构建一个自定义转储捕获内核以满足特定需求。

下面描述了激活 kdump 支持需要对系统和转储捕获内核进行的配置。

系统内核配置选项

----

1) 选中"Processor type and features"下的"kexec system call"

```
CONFIG_KEXEC=y
```

2) 选中"Filesystem" -> "Pseudo filesystems"下的"sysfs file system support"。该项缺省为选中

```
CONFIG_SYSFS=y
```

注意如果没有选中"General Setup"下的"**Configure standard kernel features (for small systems)**"，那么在"Pseudo filesystems"菜单下可能看不到"sysfs file system support"。这时，检查.config 文件来确保选中 sysfs：

```
grep 'CONFIG_SYSFS' .config
```

3) 选中"Kernel hacking"下的"Compile the kernel with debug info"

```
CONFIG_DEBUG_INFO=Y
```

这样构建的内核会包含调试符号。转储分析工具需要一个包含调试符号的 vmlinux，以便读取并分析转储文件

转储捕获内核配置选项(独立于体系结构)

----

1) 选中"Processor type and features"下的"kernel crash dumps"

```
CONFIG_CRASH_DUMP=y
```

2) 选中"Filesystems" -> "Pseudo filesystems"下的"/proc/vmcore support"

```
CONFIG_PROC_VMCORE=y
```

(当选中 CONFIG_CRASH_DUMP 时缺省会设置 CONFIG_PROC_VMCORE)

转储捕获内核配置选项(依赖于体系结构，i386 和 x86_64)

----

1) 在 i386 机器上，在"Processor type and features"下选中高端内存支持：

```
CONFIG_HIGHMEM64G=y
```

或者

```
CONFIG_HIGHMEM4G
```

2) 在 i386 和 x86_64 机器上，关闭"Processor type and features"下的 SMP 支持：

```
CONFIG_SMP=n
```

（如果 CONFIG_SMP=y，那么加载转储捕获内核时在内核命令行中指定 **maxcpus=1**。参考章节“加载转储捕获内核”）

3) 如果要构建使用可重定位内核，选中"Processor type and features"下的"Build a relocatable kernel"：

```
CONFIG_RELOCATABLE=y
```

4) 为"Physical address where the kernel is loaded"(在"Processor type and features"菜单下)设定适当的值。只有选中"kernel crash dumps"才能看到该选项。使用的值取决于内核是否是可重定位的

如果使用的是可重定位内核，则使用

CONFIG_PHYSICAL_START=0x100000

这样编译的内核使用的物理地址为 **1MB**，但是内核时可重定位的，它在任何地址处都可以运行，这样 kexec bootloader 可以将它加载到用于转储捕获内核的内存区域

否则使用引导参数"**crashkernel=Y@X**"指定用于第二个内核的内存区域。这里 X

是用于转储捕获内核的内存起始位置，通常为 **16MB(0x1000000)**。所以可以设置

CONFIG_PHYSICAL_START=0x1000000

5) 编译并安装内核及其模块。不要将该内核添加到 boot loader 的配置文件中

转储捕获内核配置选项(依赖于体系结构，ppc64)

----

1) 选中"Kernel"选项下的"Build a kdump crash kernel"

```
CONFIG_CRASH_DUMP=y
```

2) 选中"Build a relocatable kernel"

```
CONFIG_RELOCATABLE=y
```

构建并安装内核及其模块

转储捕获内核配置选项(依赖于体系结构，ia64)

- 除去上面体系机构相关那一节指定的选项，构建 ia64 架构上的转储捕获内核不需要特定选项。这样可以在需要时将系统内核用作转储捕获内核

  可以在系统内核运行时自动设置 crashkernel 区域。可以通过指定基址为 0 或者忽略做到这一点：

```
crashkernel=256M@0
```

或者

```
crashkernel=256M
```

如果指定了起始地址，注意地址地址为 64Mb 对齐，所以如果起始地址不符合对齐要求，那么不符合对齐要求的那部分空间就被浪费掉了

crashkernel 扩展语法

====================

虽然"**crashkernel=size[@offset]**"语法适用于多数配置，但有时候根据系统 RAM 值来设定保留内存更方便——主要用于发行版预先设置内核参数以避免机器中移除一部分内存后导致系统不可引导。

语法如下：

```
crashkernel=<range1>:<size1>[,<range2>:<size2>,...][@offset]
range=start-[end]
```

'start'包含在内，但不包含'end'。

示例：

```
crashkernel=512M-2G:64M,2G-:128M
```

解析：

1) 如果 RAM 不足 512M，那么不保留内存(用于"rescue"的情况)

2) 如果 RAM 大于 512M 但不足 2G（不包含 2G），那么保留 64M

3) 如果 RAM 大于 2G，那么保留 128M

启动进入系统内核

================

1) 更新 boot loader(比如 grub、yaboot 或者 lilo)的配置文件

2) 启动系统内核，指定引导参数"**crashkernel=Y@X**"，这里 Y 指定了用于转储捕获内核的内存区域大小，X 指定了该保留区域的起始位置。例如，"crashkernel=64M@16M" 让系统内核将物理地址 0x01000000(16MB)起的 64MB 内存保留下来用于转储捕获内核

在 x86 和 x86_64 机器上使用"**crashkernel=64M@16M**"

在 ppc64 机器上使用"**crashkernel=128M@32M**"

在 ia64 机器上通常使用 **256M@256M**。在 ia64 机器上可以动态设置该区域，参考上面关于转储捕获内核配置选项的描述

加载转储内核

============

启动系统内核后，需要加载转储捕获内核。根据体系结构和映像的类型（是否可重定位）的不同，可以加载转储捕获内核的未压缩的 vmlinux 或者压缩的 bzImage/vmlinuz。下面是总结：

i386 和 x86_64 机器

- 如果内核不可重定位则使用 vmlinux
- 如果内核可重定位则使用 bzImage/vmlinuz

ppc64 机器

- 使用 vmlinux

ia64 机器

- 使用 vmlinux 或者 vmlinuz.gz

如果使用未压缩的 vmlinux 映像，则使用下列命令加载转储捕获内核：

```
kexec -p <dump-capture-kernel-vmlinux-image> \
--initrd=<initrd-for-dump-capture-kernel> --args-linux \
--append="root=<root-dev> <arch-specific-options>"
```

如果使用压缩的 bzImage/vmlinuz，使用下列命令加载转储捕获内核：

```
kexec -p <dump-capture-kernel-bzImage> \
--initrd=<initrd-for-dump-capture-kernel> \
--append="root=<root-dev> <arch-specific-options>"
```

注意，不需要为 ia64 指定--args-linux。本来打算将其针对 ia64 设置为 no-op，现在忽略不用。

下面是加载转储捕获内核时对应不同体系机构使用的命令行参数：

i386、x86_64 和 ia64

```
"1 irqpoll maxcpus=1 reset_devices"
```

ppc64

```
"1 maxcpus=1 noirqdistrib reset_devices"
```

加载转储捕获内核时要注意：

* 缺省，为支持多于 4GB 的内存，ELF header 以 ELF64 格式存放。在 i386 机器上，kexec 自动检查是否物理内存多于 4GB，如果不是则使用 ELF32。所以在未开启 PAE 的系统上总是使用 ELF32

  可以使用**--elf32-core-headers**选项强制生成 ELF32 headers。有必要使用该选项，因为当前在 32 位系统上 GDB 并不能打开使用 ELF64 header 的 vmcore 文件

* 引导参数"**irqpoll**"减少了因为在转储捕获内核中共享中断导致的驱动程序初始化失败的情况

* 在格式中指定的**<root-dev>**必须对应 mount 命令输出中的 root 设备名

* 引导参数"1"会引导转储捕获内核到无网络的单用户状态；如果需要使用网络，使用"**3**"

* 通常不需要启动一个 SMP 内核来捕获转储。因此通常构建一个 UP 转储捕获内核或者加载转储捕获内核时指定选项 **maxcpus=1**

内核 Panic

==========

如前所述成功加载转储捕获内核后，如果系统崩溃那么重启会进入转储捕获内核。可以通过 panic()、die()、die_nmi() 和 sysrq 处理器(ALT-SysRq-c)触发崩溃。

下列情形下执行触发崩溃操作：

1) 如果检测到 hard lockup 并且配置了"NMI watchdog"，系统会启动进入转储捕获内核 (die_nmi())

2) 如果调用 die()，而恰巧当前进程的 PID 为 0 或者 1，或者在中断上下文中调用 die()，或者调用 die()并且设置了 panic_on_oops，系统会启动进入转储捕获内核

3) 在 powerpc 系统上，当产生 soft-reset 时，所有的 CPU 都会调用 die()，系统会启动进入转储捕获内核

4) 为测试而通过"ALT-SysRq-c"、"echo c > /proc/sysrq-trigger"或者编写一个模块强制产生 panic，会触发崩溃

写出到转储文件

==============

启动转储捕获内核后，使用下列命令写入转储文件：

```
cp /proc/vmcore <dump-file>
```

还可以通过设备/dev/oldmem 访问转储内存，以获取线性的 raw 数据。使用下面的命令创建设备：

```
mknod /dev/oldmem c 1 12
```

使用 dd 命令，加上适当的 count、bs 参数，跳到转储的指定部分。

要查看整个内存，使用下面的命令：

```
dd if=/dev/oldmem of=oldmem.001
```

分析

====

在分析转储映像之前，要启动到稳定内核。

使用 GDB 可以对从/proc/vmcore 复制得到的转储文件进行有限的分析。使用编译时指定-g 选项的 vmlinux，执行下面的命令：

```
gdb vmlinux <dump-file>
```

可以进行 0 号处理器上任务的栈跟踪、寄存器和内存内容的显示。

注意：GDB 不能分析 x86 平台上产生的 ELF64 格式的转储文件。在一个最大内存为 4GB 的系统上，可以通过给转储捕获内核指定选项**--elf32-core-headers** 来产生 ELF32 格式的 header。

还可以使用 crash 工具分析 kdump 格式的转储文件。可以通过下面的 URL 访问 Dave Anderson 个人站点以获取 crash：

```
http://people.redhat.com/~anderson/
```

To Do

=====

1) 使所有体系结构都支持可重定位内核，这样可以让多种内核支持崩溃转储机制；使用相同的内核作为系统内核来捕获转储

联系信息

========

Vivek Goyal (vgoyal@in.ibm.com)

Maneesh Soni (maneesh@in.ibm.com)

【原文】kdump.txt

```
================================================================
Documentation for Kdump - The kexec-based Crash Dumping Solution
================================================================

This document includes overview, setup and installation, and analysis
information.

Overview
========

Kdump uses kexec to quickly boot to a dump-capture kernel whenever a
dump of the system kernel's memory needs to be taken (for example, when
the system panics). The system kernel's memory image is preserved across
the reboot and is accessible to the dump-capture kernel.

You can use common commands, such as cp and scp, to copy the
memory image to a dump file on the local disk, or across the network to
a remote system.

Kdump and kexec are currently supported on the x86, x86_64, ppc64 and ia64
architectures.

When the system kernel boots, it reserves a small section of memory for
the dump-capture kernel. This ensures that ongoing Direct Memory Access
(DMA) from the system kernel does not corrupt the dump-capture kernel.
The kexec -p command loads the dump-capture kernel into this reserved
memory.

On x86 machines, the first 640 KB of physical memory is needed to boot,
regardless of where the kernel loads. Therefore, kexec backs up this
region just before rebooting into the dump-capture kernel.

Similarly on PPC64 machines first 32KB of physical memory is needed for
booting regardless of where the kernel is loaded and to support 64K page
size kexec backs up the first 64KB memory.

All of the necessary information about the system kernel's core image is
encoded in the ELF format, and stored in a reserved area of memory
before a crash. The physical address of the start of the ELF header is
passed to the dump-capture kernel through the elfcorehdr= boot
parameter.

With the dump-capture kernel, you can access the memory image, or "old
memory," in two ways:

- Through a /dev/oldmem device interface. A capture utility can read the
  device file and write out the memory in raw format. This is a raw dump
  of memory. Analysis and capture tools must be intelligent enough to
  determine where to look for the right information.

- Through /proc/vmcore. This exports the dump as an ELF-format file that
```

you can write out using file copy commands such as cp or scp. Further, you can use analysis tools such as the GNU Debugger (GDB) and the Crash tool to debug the dump file. This method ensures that the dump pages are correctly ordered.

Setup and Installation

Install kexec-tools

1) Login as the root user.

2) Download the kexec-tools user-space package from the following URL:

http://www.kernel.org/pub/linux/kernel/people/horms/kexec-tools/kexec-tools.tar.gz

This is a symlink to the latest version.

The latest kexec-tools git tree is available at:

git://git.kernel.org/pub/scm/linux/kernel/git/horms/kexec-tools.git
or
http://www.kernel.org/git/?p=linux/kernel/git/horms/kexec-tools.git

More information about kexec-tools can be found at
http://www.kernel.org/pub/linux/kernel/people/horms/kexec-tools/README.html

3) Unpack the tarball with the tar command, as follows:

```
   tar xvpzf kexec-tools.tar.gz
```

4) Change to the kexec-tools directory, as follows:

```
   cd kexec-tools-VERSION
```

5) Configure the package, as follows:

```
   ./configure
```

6) Compile the package, as follows:

```
   make
```

7) Install the package, as follows:

```
   make install
```

Build the system and dump-capture kernels

There are two possible methods of using Kdump.

1) Build a separate custom dump-capture kernel for capturing the kernel core dump.

2) Or use the system kernel binary itself as dump-capture kernel and there is no need to build a separate dump-capture kernel. This is possible only with the architectures which support a relocatable kernel. As of today, i386, x86_64, ppc64 and ia64 architectures support relocatable kernel.

Building a relocatable kernel is advantageous from the point of view that one does not have to build a second kernel for capturing the dump. But at the same time one might want to build a custom dump capture kernel suitable to his needs.

Following are the configuration setting required for system and dump-capture kernels for enabling kdump support.

System kernel config options

1) Enable "kexec system call" in "Processor type and features."

```
CONFIG_KEXEC=y
```

2) Enable "sysfs file system support" in "Filesystem" -> "Pseudo filesystems." This is usually enabled by default.

```
CONFIG_SYSFS=y
```

Note that "sysfs file system support" might not appear in the "Pseudo filesystems" menu if "Configure standard kernel features (for small systems)" is not enabled in "General Setup." In this case, check the .config file itself to ensure that sysfs is turned on, as follows:

```
grep 'CONFIG_SYSFS' .config
```

3) Enable "Compile the kernel with debug info" in "Kernel hacking."

```
CONFIG_DEBUG_INFO=Y
```

This causes the kernel to be built with debug symbols. The dump analysis tools require a vmlinux with debug symbols in order to read and analyze a dump file.

Dump-capture kernel config options (Arch Independent)

1) Enable "kernel crash dumps" support under "Processor type and features":

```
CONFIG_CRASH_DUMP=y
```

2) Enable "/proc/vmcore support" under "Filesystems" -> "Pseudo filesystems".

```
   CONFIG_PROC_VMCORE=y
   (CONFIG_PROC_VMCORE is set by default when CONFIG_CRASH_DUMP is selected.)
```

Dump-capture kernel config options (Arch Dependent, i386 and x86_64)

1) On i386, enable high memory support under "Processor type and features":

```
   CONFIG_HIGHMEM64G=y
   or
   CONFIG_HIGHMEM4G
```

2) On i386 and x86_64, disable symmetric multi-processing support under "Processor type and features":

```
   CONFIG_SMP=n
```

   (If CONFIG_SMP=y, then specify maxcpus=1 on the kernel command line when loading the dump-capture kernel, see section "Load the Dump-capture Kernel".)

3) If one wants to build and use a relocatable kernel, Enable "Build a relocatable kernel" support under "Processor type and features"

```
   CONFIG_RELOCATABLE=y
```

4) Use a suitable value for "Physical address where the kernel is loaded" (under "Processor type and features"). This only appears when "kernel crash dumps" is enabled. A suitable value depends upon whether kernel is relocatable or not.

   If you are using a relocatable kernel use CONFIG_PHYSICAL_START=0x100000 This will compile the kernel for physical address 1MB, but given the fact kernel is relocatable, it can be run from any physical address hence kexec boot loader will load it in memory region reserved for dump-capture kernel.

   Otherwise it should be the start of memory region reserved for second kernel using boot parameter "crashkernel=Y@X". Here X is start of memory region reserved for dump-capture kernel. Generally X is 16MB (0x1000000). So you can set CONFIG_PHYSICAL_START=0x1000000

5) Make and install the kernel and its modules. DO NOT add this kernel to the boot loader configuration files.

Dump-capture kernel config options (Arch Dependent, ppc64)

1) Enable "Build a kdump crash kernel" support under "Kernel" options:

```
   CONFIG_CRASH_DUMP=y
```

2)   Enable "Build a relocatable kernel" support

```
   CONFIG_RELOCATABLE=y
```

   Make and install the kernel and its modules.

Dump-capture kernel config options (Arch Dependent, ia64)

- No specific options are required to create a dump-capture kernel
  for ia64, other than those specified in the arch independent section
  above. This means that it is possible to use the system kernel
  as a dump-capture kernel if desired.

  The crashkernel region can be automatically placed by the system
  kernel at run time. This is done by specifying the base address as 0,
  or omitting it all together.

```
  crashkernel=256M@0
  or
  crashkernel=256M
```

  If the start address is specified, note that the start address of the
  kernel will be aligned to 64Mb, so if the start address is not then
  any space below the alignment point will be wasted.

Extended crashkernel syntax

While the "crashkernel=size[@offset]" syntax is sufficient for most
configurations, sometimes it's handy to have the reserved memory dependent
on the value of System RAM -- that's mostly for distributors that pre-setup
the kernel command line to avoid a unbootable system after some memory has
been removed from the machine.

The syntax is:

```
    crashkernel=<range1>:<size1>[,<range2>:<size2>,...][@offset]
    range=start-[end]
```

    'start' is inclusive and 'end' is exclusive.

For example:

```
    crashkernel=512M-2G:64M,2G-:128M
```

This would mean:

```
   1) if the RAM is smaller than 512M, then don't reserve anything
      (this is the "rescue" case)
   2) if the RAM size is between 512M and 2G (exclusive), then reserve 64M
   3) if the RAM size is larger than 2G, then reserve 128M
```

Boot into System Kernel

1) Update the boot loader (such as grub, yaboot, or lilo) configuration
   files as necessary.

2) Boot the system kernel with the boot parameter "crashkernel=Y@X",
   where Y specifies how much memory to reserve for the dump-capture kernel
   and X specifies the beginning of this reserved memory. For example,
   "crashkernel=64M@16M" tells the system kernel to reserve 64 MB of memory
   starting at physical address 0x01000000 (16MB) for the dump-capture kernel.

   On x86 and x86_64, use "crashkernel=64M@16M".

   On ppc64, use "crashkernel=128M@32M".

   On ia64, 256M@256M is a generous value that typically works.
   The region may be automatically placed on ia64, see the
   dump-capture kernel config option notes above.

Load the Dump-capture Kernel

After booting to the system kernel, dump-capture kernel needs to be
loaded.

Based on the architecture and type of image (relocatable or not), one
can choose to load the uncompressed vmlinux or compressed bzImage/vmlinuz
of dump-capture kernel. Following is the summary.

```
For i386 and x86_64:
	- Use vmlinux if kernel is not relocatable.
	- Use bzImage/vmlinuz if kernel is relocatable.
For ppc64:
	- Use vmlinux
For ia64:
	- Use vmlinux or vmlinuz.gz
```

If you are using a uncompressed vmlinux image then use following command
to load dump-capture kernel.

```
   kexec -p <dump-capture-kernel-vmlinux-image> \
   --initrd=<initrd-for-dump-capture-kernel> --args-linux \
   --append="root=<root-dev> <arch-specific-options>"
```

If you are using a compressed bzImage/vmlinuz, then use following command to load dump-capture kernel.

```
   kexec -p <dump-capture-kernel-bzImage> \
   --initrd=<initrd-for-dump-capture-kernel> \
   --append="root=<root-dev> <arch-specific-options>"
```

Please note, that --args-linux does not need to be specified for ia64. It is planned to make this a no-op on that architecture, but for now it should be omitted

Following are the arch specific command line options to be used while loading dump-capture kernel.

For i386, x86_64 and ia64:

```
        "1 irqpoll maxcpus=1 reset_devices"
```

For ppc64:

```
        "1 maxcpus=1 noirqdistrib reset_devices"
```

Notes on loading the dump-capture kernel:

* By default, the ELF headers are stored in ELF64 format to support systems with more than 4GB memory. On i386, kexec automatically checks if the physical RAM size exceeds the 4 GB limit and if not, uses ELF32. So, on non-PAE systems, ELF32 is always used.

  The --elf32-core-headers option can be used to force the generation of ELF32 headers. This is necessary because GDB currently cannot open vmcore files with ELF64 headers on 32-bit systems.

* The "irqpoll" boot parameter reduces driver initialization failures due to shared interrupts in the dump-capture kernel.

* You must specify <root-dev> in the format corresponding to the root device name in the output of mount command.

* Boot parameter "1" boots the dump-capture kernel into single-user mode without networking. If you want networking, use "3".

* We generally don' have to bring up a SMP kernel just to capture the dump. Hence generally it is useful either to build a UP dump-capture kernel or specify maxcpus=1 option while loading dump-capture kernel.

Kernel Panic

After successfully loading the dump-capture kernel as previously described, the system will reboot into the dump-capture kernel if a system crash is triggered. Trigger points are located in panic(), die(), die_nmi() and in the sysrq handler (ALT-SysRq-c).

The following conditions will execute a crash trigger point:

If a hard lockup is detected and "NMI watchdog" is configured, the system will boot into the dump-capture kernel (die_nmi()).

If die() is called, and it happens to be a thread with pid 0 or 1, or die() is called inside interrupt context or die() is called and panic_on_oops is set, the system will boot into the dump-capture kernel.

On powerpc systems when a soft-reset is generated, die() is called by all cpus and the system will boot into the dump-capture kernel.

For testing purposes, you can trigger a crash by using "ALT-SysRq-c", "echo c > /proc/sysrq-trigger" or write a module to force the panic.

Write Out the Dump File

After the dump-capture kernel is booted, write out the dump file with the following command:

```
   cp /proc/vmcore <dump-file>
```

You can also access dumped memory as a /dev/oldmem device for a linear and raw view. To create the device, use the following command:

```
    mknod /dev/oldmem c 1 12
```

Use the dd command with suitable options for count, bs, and skip to access specific portions of the dump.

To see the entire memory, use the following command:

```
   dd if=/dev/oldmem of=oldmem.001
```

Analysis

Before analyzing the dump image, you should reboot into a stable kernel.

You can do limited analysis using GDB on the dump file copied out of /proc/vmcore. Use the debug vmlinux built with -g and run the following command:

```
   gdb vmlinux <dump-file>
```

Stack trace for the task on processor 0, register display, and memory display work fine.

Note: GDB cannot analyze core files generated in ELF64 format for x86. On systems with a maximum of 4GB of memory, you can generate ELF32-format headers using the --elf32-core-headers kernel option on the dump kernel.

You can also use the Crash utility to analyze dump files in Kdump format. Crash is available on Dave Anderson's site at the following URL:

   http://people.redhat.com/~anderson/

To Do

1) Provide relocatable kernels for all architectures to help in maintaining multiple kernels for crash_dump, and the same kernel as the system kernel can be used to capture the dump.

Contact

Vivek Goyal (vgoyal@in.ibm.com)
Maneesh Soni (maneesh@in.ibm.com)