The Taming of the Kernel Dump

Presenter: Petr Tesařík
Date: 14th November 2023
Location: LPC Richmond, VA
Background
Motivation

- A wheel that was reinvented too many times:
  - lcrash
Motivation

- A wheel that was reinvented too many times:
  - lcrash
  - crash
Motivation

• A wheel that was reinvented too many times:
  – lcrash
  – crash
  – makedumpfile
Motivation

- A wheel that was reinvented too many times:
  - lcrash
  - crash
  - makedumpfile
  - (kdumpid)
Motivation

- A wheel that was reinvented too many times:
  - lcrash
  - crash
  - makedumpfile
  - (kdumpid)

- Let’s make THE wheel! Reliable. Universal. One wheel to rule them all.
Motivation

• A wheel that was reinvented too many times:
  – lcrash
  – crash
  – makedumpfile
  – (kdumpid)

• Let’s make THE wheel! Reliable. Universal.
  – One wheel to rule them all.
Design Goals

- Mimic actual hardware
- Cross-platform
- Support non-Linux operating systems
- Small enough for crash kernel
- Fast enough to run on crashed systems
- Expose detailed information
- Reasonably easy to use
Challenges
## Architecture Zoo

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Crash (crash)</th>
<th>makedumpfile</th>
<th>libkdumpfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>aarch64 (arm64)</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>arm</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>ia32 (x86)</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>ia64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>loongarch64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mips</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mips64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>powerpc</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>ppc64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>riscv64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>s390</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>s390x</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>sparc64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>x86_64</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Architecture</td>
<td>crash</td>
<td>makedumpfile</td>
<td>libkdumpfile</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>alpha</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>aarch64 (arm64)</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>arm</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>ia32 (x86)</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>ia64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>loongarch64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>mips</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>mips64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>powerpc</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>ppc64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>riscv64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>s390</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>s390x</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>sparc64</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>x86_64</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>
Adding Architectures

- There is a HOWTO:
  https://github.com/ptesarik/libkdumpfile/wiki/Adding-Architectures
  - Page Table Translation (or whatever the hardware does)
  - Linux Support
  - CPU Registers
Format Zoo

• Common
  - bare metal: ELF, kdump (makedumpfile)
  - virtual: qemu, libvirt, Xen ELF, xc_core, xc_save

• Vendor-specific
  - SADUMP, Ramdump, S390DUMP, z/VM vmdump

• Historic
  - LKCD, diskdump
Adding Formats

- No HOWTO document :-(
- Requires good knowledge of the internal file cache API
- Some formats may need non-existent infrastructure
- Dump formats often poorly documented
  - KDUMP is defined by makedumpfile sources
  - SADUMP is non-public and reverse-engineered
Conversion PoC

- Makedumpfile once partially converted
- v1.6.8 (~100 commits behind now)
  - File reads: easy, some fallouts (e.g. cache stats)
  - Address translation: mostly fine (except SADUMP and Xen)
  - VMCOREINFO: somewhat messy
- Complies with current Linux development trends:
  19 files changed, 741 insertions(+), 4401 deletions(-)
Other Wild Ideas
GDB Server

- https://github.com/ptesarik/kdump-gdbserver
  - gdbserver frontend to crash dump files
  - written in Python using pykdumpfile bindings
  - assumes distinct kernel-space and user-space addresses
Write Support

- How to design the API?
  - Some preparatory work (CoW clone of the attribute hierarchy)
  - Replace makedumpfile input and output routines?
    - Initialize input context
    - Adjust file.pagemap and file.format of a cloned context
    - Write output context
  - What about split dump files?
Thank you!