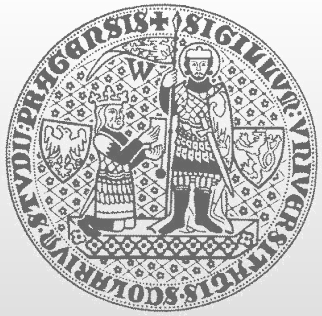
crash

Crash Dump Analysis 2014/2015



CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Department of Distributed and Dependable Systems

D3S

ORACLE®

SUSE®

redhat®

crash – introduction

- `crash`: the tool of choice for Linux crash dumps
  - Created by David Anderson from Red Hat
  - Understands all dump formats – kdump (compressed), netdump, diskdump, xendump, KVM dump, s390, LKCD
  - Understands some kernel internals: memory mapping, tasks, SLAB objects, …
  - Can e.g. walk linked lists, pipe output for further postprocessing,
  - Extensible with Eppic – a C intepreter tailored to work with C structures stored in a dump
    - `http://code.google.com/p/eppic/wiki/README`

crash – disadvantages

- crash has also disadvantages...
  - Uses gdb internally, but mostly just invokes some gdb query and pipes its output
  - Backtraces are not like gdb has with debuginfo
  - Some things are done both in crash and gdb
    - The codebase is hard to maintain
  - Machine running crash must be of same architecture as the dump

crash – possible improvements

- There are some efforts at SUSE to improve crash
  - E.g. allow DWARF debuginfo based backtraces
  - Python scripting in crash instead of Eppic
- One bigger idea is to return to pure gdb, add kdump format support and use Python extensions where possible for kernel knowledge
  - No architecture limitation
  - Nice backtraces by gdb itself
  - Complex automated analysis possible using Python on top

crash – libkdumpfile approach

- The effort resulted in a new library libkdumpfile that can be plugged to gdb
  - Handles kdump formats and address translation
  - Processes mapped to gdb threads, backtraces done by gdb with debuginfo
  - Kernel entities can be accessed by Python
  - Lots of corner cases and functionality missing
- Still a long way to go before it can replace traditional crash

Invoking crash

- On core dump
  - `crash vmlinux.gz vmlinux.debug vmcore`
- On live system
  - `crash vmlinux.gz vmlinux.debug`
- Options
  - `-s` silent, output not paged to `less`
  - `-i file` execute commands from file
  - `--mod dir` search for module debuginfo in dir
  - `--minimal` restrict to basic commands for broken dumps

Invoking crash – welcome screen

```
      KERNEL: vmlinux.gz
   DEBUGINFO: vmlinux.debug
    DUMPFILE: vmcore
        CPUS: 8
        DATE: Thu Apr 10 16:07:34 2014
      UPTIME: 7 days, 03:17:51
LOAD AVERAGE: 0.01, 0.02, 0.05
       TASKS: 161
    NODENAME: lpapp114
     RELEASE: 3.0.101-0.7.17-default
     VERSION: #1 SMP Tue Feb 4 13:24:49 UTC 2014 (90aac76)
     MACHINE: x86_64  (2399 Mhz)
      MEMORY: 64 GB
       PANIC: "[615702.371868] kernel BUG at
/usr/src/packages/BUILD/kernel-default-3.0.101/linux-
3.0/mm/slab.c:539!"
         PID: 58
     COMMAND: "kworker/6:1"
        TASK: ffff88080e03e680  [THREAD_INFO: ffff88080e040000]
         CPU: 6
       STATE: TASK_RUNNING (PANIC)
```

Invoking crash – help screen

```
crash> help

*               files           mach            repeat          timer
alias           foreach         mod             runq            tree
ascii           fuser           mount           search          union
bt              gdb             net             set             vm
btop            help            p               sig             vtop
dev             ipcs            ps              struct          waitq
dis             irq             pte             swap            whatis
eval            kmem            ptob            sym             wr
exit            list            ptov            sys             q
extend          log             rd              task

crash version: 7.1.0    gdb version: 7.6
For help on any command above, enter "help <command>".
For help on input options, enter "help input".
For help on output options, enter "help output".
```

crash – setting up the session

- `mod -s module [objfile]` – load symbols and debuginfo for given module
  - searches `/lib/modules/<release>` and then directory of `vmlinux` given as param
- `mod -S [directory]` – load all modules that were in use in the crashed kernel
- `set scroll [on|off] / less / more /` CRASHPAGER – pager adjustment
- `set [radix [10|16]] [hex|dec]` – for numbers output
- `set offline [show | hide]` – show offline CPUs?

Basic crash commands

- `dmesg (log)` – same as the shell command
- `mount` – filesystems including vfsmount and superblock pointers
  - `-f` – dentries and inodes for open files (<3.13)
  - can be limited to single fs by relevant object addr
- `swap` – swap device info
- `fuser [path|inode]` – list tasks using a file
- `mach [-m | -c]` – machine specific data (mem, cpu)
- `dev [-d]` – device data (I/O statistics)
- `net [-s|-S]` – network related info (sockets details)
- `sys [-t] [config]` – system data / taint flags / config options
- `mod -t [mod]` – module taint flags

Listing processes with ps

```
crash> ps
   PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
>     0      0   0  ffffffff81a0b020  RU   0.0       0      0  [swapper]
>     0      2   1  ffff88080f442240  RU   0.0       0      0  [kworker/0:0]
>     0      2   2  ffff88080f47e380  RU   0.0       0      0  [kworker/0:1]
>     0      2   3  ffff88080f48a400  RU   0.0       0      0  [kworker/0:1]
>     0      2   4  ffff88080f4a8540  RU   0.0       0      0  [kworker/0:1]
>     0      2   5  ffff88080f4b45c0  RU   0.0       0      0  [kworker/0:1]
      0      2   6  ffff88080f4ee080  RU   0.0       0      0  [kworker/0:1]
>     0      2   7  ffff88080f4fc100  RU   0.0       0      0  [kworker/0:1]
      1      0   3  ffff88080f868040  IN   0.0   10528    840  init
```

- `ps [pid|task|command] [-k|-u|-G]`
  - Filter tasks; kernel/user/thread group leader only
- `ps [-p|-c]` – parental/children hierarchy
- `ps [-a|-r]` – cmdline args+env / rlimits

Listing processes with ps

- `ps -S` – summary of task states

```
crash> ps -S
  RU: 5
  IN: 259
  UN: 31
  ZO: 1
```

- `ps -l` – sort by last run timestamp (desc.)
  - `ps -m` – similar but with human readable time
  - `-C [x,y-z]` – restrict/group by CPU's

```
crash> ps -l
[615702371793211] [IN]  PID: 1578   TASK: ffff88100df62300  CPU: 4   COMMAND: "kworker/u:4"
[615702371790715] [RU]  PID: 58     TASK: ffff88080e03e680  CPU: 6   COMMAND: "kworker/6:1"
[615702351818816] [IN]  PID: 2147   TASK: ffff88080bf5e140  CPU: 0   COMMAND: "xfsaild/dm-0"

crash> ps -m
[0 00:00:00.000] [IN]  PID: 1578   TASK: ffff88100df62300  CPU: 4   COMMAND: "kworker/u:4"
[0 00:00:00.000] [RU]  PID: 58     TASK: ffff88080e03e680  CPU: 6   COMMAND: "kworker/6:1"
[0 00:00:00.000] [IN]  PID: 2147   TASK: ffff88080bf5e140  CPU: 0   COMMAND: "xfsaild/dm-0"
```

Listing CPU run queues with runq

```
crash> runq
CPU 0 RUNQUEUE: ffff88083fa11180
  CURRENT: PID: 0      TASK: ffffffff81a0b020  COMMAND: "swapper"
  RT PRIO_ARRAY: ffff88083fa11310
     [no tasks queued]
  CFS RB_ROOT: ffff88083fa11220
     [no tasks queued]
```

- `runq -t` – timestamps for CPU+tasks

```
crash> runq -t
 CPU 0: 615702351824321
        0000000000000000  PID: 0      TASK: ffffffff81a0b020  COMMAND: "swapper"
 CPU 1: 615702068264343
        0000000000000000  PID: 0      TASK: ffff88080f442240  COMMAND: "kworker/0:0"
 CPU 2: 615702371790715
        615702371790715   PID: 58     TASK: ffff88080e03e680  COMMAND: "kworker/2:1"
```

- `runq -m` – task uptime

```
crash> runq -m
 CPU 0: [7 03:01:42.351]  PID: 0      TASK: ffffffff81a0b020  COMMAND: "swapper"
 CPU 1: [7 03:01:42.068]  PID: 0      TASK: ffff88080f442240  COMMAND: "kworker/0:0"
 CPU 2: [0 00:00:00.000]  PID: 58     TASK: ffff88080e03e680  COMMAND: "kworker/2:1"
```

Backtraces

- `bt [task|pid] [-a]` – show backtrace(s)

```
crash> bt
PID: 58     TASK: ffff88080e03e680  CPU: 6   COMMAND: "kworker/6:1"
 #0 [ffff88080e041ad0] machine_kexec at ffffffff810267ae
 #1 [ffff88080e041b20] crash_kexec at ffffffff810a42da
 #2 [ffff88080e041bf0] oops_end at ffffffff8144ab28
 #3 [ffff88080e041c10] do_invalid_op at ffffffff810035f4
 #4 [ffff88080e041cb0] invalid_op at ffffffff8145233b
    [exception RIP: free_block+122]
    RIP: ffffffff8113d98a  RSP: ffff88080e041d60  RFLAGS: 00010046
    RAX: 0060000000000000  RBX: ffff880a0db0d2c0  RCX: ffff88081080ce40
    RDX: ffffea00232fead8  RSI: ffff88080de29040  RDI: ffff880a0db0d2c0
    RBP: ffff88080f4c7838   R8: 0000000000000000   R9: ffff88080d270ef0
    R10: ffff88080d270ef0  R11: ffff88080d270c70  R12: 0000000000000004
    R13: ffff88100f920100  R14: 0000000000000018  R15: ffffea0000000000
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
 #5 [ffff88080e041d98] drain_array at ffffffff8113dcc4
 #6 [ffff88080e041dd8] cache_reap at ffffffff8113e53e
 #7 [ffff88080e041e28] process_one_work at ffffffff81074b2c
 #8 [ffff88080e041e78] worker_thread at ffffffff810776ca
 #9 [ffff88080e041ee8] kthread at ffffffff8107ba36
#10 [ffff88080e041f48] kernel_thread_helper at ffffffff814524c4
```

Backtraces

- `bt -l` – include file:line translation
- `bt -s` – include offset in symbol (radix `-x`/`-d`)
- `bt -t` – all text symbols (when unwind fails)
- `bt -T` – same but whole stack, not from RSP
- `[foreach] bt -R [sym|addr]` – filter by content on stack
- `bt -f` – include full raw stack content
- `bt -F[F]` – like `-f`, but translate to symbols or slab objects where possible (`-FF` adds addr)

Backtraces – example with -FF

```
crash> bt -FFsx  # (“set radix” is 10)
(...)
 #5 [ffff88080e041d98] drain_array+0xa4 at ffffffff8113dcc4
    ffff88080e041da0: [ffff88081080ce80:size-128] [ffff88081080ce40:size-128]
    ffff88080e041db0: [ffff88081080cec0:size-128] [ffff88100f920100:kmem_cache]
    ffff88080e041dc0: 0000000000000001 ffff88083fa74605
    ffff88080e041dd0: 0000000000000000 cache_reap+126
 #6 [ffff88080e041dd8] cache_reap+0x7e at ffffffff8113e53e
    ffff88080e041de0: ffff88083fa6e240 000000000000100c
    ffff88080e041df0: ffff88083fa6e2c0 0000000000000000
    ffff88080e041e00: ffff88083fa6e2c0 [ffff88080e016ac0:size-128]
    ffff88080e041e10: ffff88083fa6cf80 ffff88083fa74605
    ffff88080e041e20: cache_reap       process_one_work+364
 #7 [ffff88080e041e28] process_one_work+0x16c at ffffffff81074b2c
    ffff88080e041e30: 0000000000000000 0000000000000000
    ffff88080e041e40: ffff88083fa74600 [ffff88080e016ac0:size-128]
    ffff88080e041e50: ffff88083fa6cf80 0000000000000008
    ffff88080e041e60: ffff88083fa6cf80 [ffff88080e016ae0:size-128]
    ffff88080e041e70: 0000000000011c00 worker_thread+378
 #8 [ffff88080e041e78] worker_thread+0x17a at ffffffff810776ca
    ffff88080e041e80: 0000000000011c00 [ffff88080e03e680:task_struct]
    ffff88080e041e90: 0000000000011c00 [ffff88080e03e680:task_struct]
    ffff88080e041ea0: 0000000000011c00 ffff88083fa6cf88
    ffff88080e041eb0: [ffff88080e016ae0:size-128] ffff88080e041f00
    ffff88080e041ec0: [ffff88080e016ac0:size-128] ffff88080f4e3d50
    ffff88080e041ed0: worker_thread    0000000000000000
    ffff88080e041ee0: 0000000000000000 kthread+150
```

Memory management inspection - kmem

- `kmem -i` – overview of used/free memory
- `kmem -V` – more details as in /proc/vmstat
- `kmem -z` – per-zone stats as in /proc/zoneinfo
- `kmem -p [-P] [addr/page*]` - page info
- `kmem -g [flags]` – list/translate page flag bits
- `kmem -n` – memory node info
- `kmem -f` – verify free lists (`-F` dumps pages)
- `kmem [-s|-S] [name]` – slab statistics / dump

Memory management inspection - kmem

- kmem [-s] addr – show info about address

```
crash> kmem -s ffff88080d270c70
CACHE             NAME                 OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE
ffff88100f940180 sigqueue                  160          0        72      3     4k
SLAB              MEMORY            TOTAL  ALLOCATED  FREE
ffff88080d270000  ffff88080d270090     24          0    24
FREE / [ALLOCATED]
   ffff88080d270c70  (cpu 6 cache)



crash> kmem ffff880a0db0d2c0
NODE
  1
ZONE  NAME        SIZE    FREE      MEM_MAP       START_PADDR  START_MAPNR
  2   Normal    8388608  7639146  ffffea001ce00000   840000000        0
AREA    SIZE  FREE_AREA_STRUCT
 10    4096k  ffff88103fb9a280
ffffea00232f4000  (ffff880a0db0d2c0 is 782nd of 1024 pages)

      PAGE        PHYSICAL      MAPPING       INDEX CNT FLAGS
ffffea00232fead8 a0db0d000                0        0  0 60000000000000
```

Memory inspection – p, eval

- `p [expr|symbol]` – print the value

```
crash> p jiffies
jiffies = $1 = 4449060284
```

- `eval` – evaluate an expression
  - Beware of using on symbols!

```
crash> eval -b 41dc065
hexadecimal: 41dc065
    decimal: 69058661
      octal: 407340145
     binary: 00000100000111011100000001100101
   bits set: 26 20 19 18 16 15 14 6 5 2 0

crash> eval jiffies
hexadecimal: ffffffff81bd1c80
    decimal: 18446744071591238784  (-2118312832)
      octal: 1777777777760157216200
     binary: 1111111111111111111111111111111110000001101111010001110010000000
```

Memory inspection – rd

- `rd [-o offset] [addr|symbol] [count]` – read/format memory contents
  - `-8/16/32/64` (bit size) `-d/-D` (decimal/unsigned)
  - `-a` – ASCII string (until nonchar); `-x` – supress
  - `-S[S]` – attempt symbol translation / with addr

```
crash> rd -8 -o 8 linux_banner 12
ffffffff81600028:  72 73 69 6f 6e 20 33 2e 30 2e 31 30       rsion 3.0.10

crash> rd -SS 0xffff88080f4c7818 8
ffff88080f4c7818:  [ffff88080e82d480:filp] [ffff88080e82d080:filp]
ffff88080f4c7828:  [ffff88080e4f72c0:filp] [ffff88080de298c0:filp]
ffff88080f4c7838:  ffff880a0db0d2c0 [ffff88080e2ae680:filp]
ffff88080f4c7848:  [ffff88080e82d880:filp] [ffff88080d71d2c0:filp]
```

Memory inspection – search

- `search [-m mask] [value|expr|sym|string]`
  - `mask` – ignore bits set in the mask
  - `[-s start] [-l len|-e end]` – limit space
  - `-w` – unsigned (hex) int instead of unsigned long
  - `-c` – (quoted) string
  - `-x count` – display also memory before/after
  - `-t` – search only stacks of processes

```
crash> search ffff88080db0d2c0
ffff88080b4ea120: ffff88080db0d2c0
ffff88080b4ea850: ffff88080db0d2c0
ffff88080b4ea9c0: ffff88080db0d2c0
ffff88080b4eab30: ffff88080db0d2c0
```

Memory translation commands

- `vtop [-c pid|taskp] addr` – translate virtual to physical addr

```
crash> eval jiffies
hexadecimal: ffffffff81bd1c80

crash> vtop ffffffff81bd1c80
VIRTUAL           PHYSICAL
ffffffff81bd1c80  1bd1c80

PML4 DIRECTORY: ffffffff81a03000
PAGE DIRECTORY: 1a05067
   PUD: 1a05ff0 => 1a09063
   PMD: 1a09068 => 100dd66063
   PTE: 100dd66e88 => 1bd1163
  PAGE: 1bd1000

  PTE     PHYSICAL  FLAGS
1bd1163   1bd1000  (PRESENT|RW|ACCESSED|DIRTY|GLOBAL)

      PAGE         PHYSICAL      MAPPING       INDEX CNT FLAGS
ffffea00000615b8   1bd1000                0        0  1 10000000000800 reserved
```

Memory translation commands

- `ptov` – the other direction

```
crash> ptov 1bd1000
VIRTUAL           PHYSICAL
ffff880001bd1000  1bd1000
```

- `btop/ptob` – remove/add phys. page offset

```
crash> ptob abcd
abcd: abcd000
crash> btop abcd123
abcd123: abcd
```

- `pte [value]` – translate a page table entry

```
crash> pte 1bd1163
  PTE     PHYSICAL  FLAGS
1bd1163   1bd1000  (PRESENT|RW|ACCESSED|DIRTY|GLOBAL)
```

Displaying structures

- `[struct|*] [-o] [-x|-d] my_struct` – display definition and member offsets

```
crash> struct -ox anon_vma
struct anon_vma {
   [0x0] struct anon_vma *root;
   [0x8] struct mutex mutex;
  [0x28] atomic_t refcount;
  [0x30] struct list_head head;
}
SIZE: 0x40
```

  - *name.member1,member2 addr – print instance

```
crash> *anon_vma.root,refcount,head ffff88080e59e120
  root = 0xffff88080cbcc860
  refcount = {
    counter = 1
  }
  head = {
    next = 0xffff88080d283878,
    prev = 0xffff88080d283878
  }
```

Displaying structures

- `struct -o addr` – apply offsets to addr

```
crash> struct -o anon_vma ffff88080e59e120
struct anon_vma {
  [ffff88080e59e120] struct anon_vma *root;
  [ffff88080e59e128] struct mutex mutex;
  [ffff88080e59e148] atomic_t refcount;
  [ffff88080e59e150] struct list_head head;
}
SIZE: 64
```

- `struct -l offset addr` – when `addr` is for an embedded member (such as `list_head`)

```
crash> struct -l anon_vma.head anon_vma.root,refcount ffff88080e59e150
  root = 0xffff88080cbcc860
  refcount = {
    counter = 1
  }
```

Traversing lists

- Let's look at the root anon_vma

```
crash> struct anon_vma.head 0xffff88080cbcc860
  head = {
    next = 0xffff88080e8d3c38,
    prev = 0xffff880809fcf1b8
  }
```

- Traverse the list by next pointer
  - Stops when circle detected or [-e endval]

```
crash> list 0xffff88080e8d3c38
ffff88080e8d3c38
ffff880809fcf1b8
ffff88080cbcc890

crash> list 0xffff88080e8d3c38 | wc -l
3
```

Traversing lists

- Print the whole `list_head`

```
crash> list -s list_head 0xffff88080e8d3c38
ffff88080e8d3c38
struct list_head {
  next = 0xffff880809fcf1b8,
  prev = 0xffff88080cbcc890
}
ffff880809fcf1b8
struct list_head {
  next = 0xffff88080cbcc890,
  prev = 0xffff88080e8d3c38
}
ffff88080cbcc890
struct list_head {
  next = 0xffff88080e8d3c38,
  prev = 0xffff880809fcf1b8
}
```

Traversing lists

- Print the structures containing `list_head`
  - We need to know that `anon_vma.head` is a linked list for `anon_vma_chain`, not `anon_vma` objects!

```
crash> kmem -s 0xffff88080e8d3c38
CACHE            NAME                 OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE
ffff880812f20ac0 anon_vma_chain            48       3242      8701    113     4k
SLAB              MEMORY            TOTAL  ALLOCATED  FREE
ffff88080e8d3000  ffff88080e8d3168     77         45    32
FREE / [ALLOCATED]
  [ffff88080e8d3c18]

crash> struct -o anon_vma_chain ffff88080e8d3c18
struct anon_vma_chain {
  [ffff88080e8d3c18] struct vm_area_struct *vma;
  [ffff88080e8d3c20] struct anon_vma *anon_vma;
  [ffff88080e8d3c28] struct list_head same_vma;
  [ffff88080e8d3c38] struct list_head same_anon_vma;
}
SIZE: 48
```

Traversing lists

- Now we can print the anon_vma_chain objs

```
crash> list -o anon_vma_chain.same_anon_vma \
           -s anon_vma_chain.vma,anon_vma,same_anon_vma -H 0xffff88080cbcc890

ffff88080e8d3c18
  vma = 0xffff880809fea648
  anon_vma = 0xffff88080cbcc860
  same_anon_vma = {
    next = 0xffff880809fcf1b8,
    prev = 0xffff88080cbcc890
  }

ffff880809fcf198
  vma = 0xffff88080b44fd78
  anon_vma = 0xffff88080cbcc860
  same_anon_vma = {
    next = 0xffff88080cbcc890,
    prev = 0xffff88080e8d3c38
  }
```

Other commands

- `vm [task]` – list memory areas of a task
- `sig [-l] [task]` – signal handling of a task / list
- `foreach [bt|vm|task|files|net|set|ps|sig|vtop]` – limited command subset for multiple tasks
- `dis [-l] [addr|sym] [count]` – disassemble
- `irq` – information about interrupts
- `timer` – information about timer queue
- `repeat [-seconds] cmd` – for live debugging
- `tree` – like list, but for radix and rb-tree structures
- `wr` – write memory (for live systems)
- `whatis [sym]` – display type info