Crash Utility Help

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

crash version: 8.0.1++   gdb version: 10.2
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".

```
NAME
  * - pointer-to short-cut

SYNOPSIS
  * (struct or union command arguments)

DESCRIPTION
  This command is a short-cut command that replaces the requirement to enter
  "struct" or "union" command names.  For details on the arguments to
  those commands, enter "help struct" or "help union".

EXAMPLES
  Dump the page structure at address c02943c0:

    crash> *page c02943c0
    struct page {
      next = 0xc0fae740,
      prev = 0xc0018fb0,
      inode = 0x0,
      offset = 0x3f000,
      next_hash = 0xc02d6310,
      count = {
        counter = 0x1
      },
      flags = 0x310,
      wait = 0xc02943d8,
      pprev_hash = 0x0,
      buffers = 0x0
    }


NAME
  files - open files

SYNOPSIS
  files [-d dentry] | [-p inode] | [-c] [-R reference] [pid | taskp] ...

DESCRIPTION
  This command displays information about open files of a context.
  It prints the context's current root directory and current working
  directory, and then for each open file descriptor it prints a pointer
  to its file struct, a pointer to its dentry struct, a pointer to the
  inode, the file type, and the pathname.  If no arguments are entered,
  the current context is used.  The -R option, typically invoked from
  "foreach files", searches for references to a supplied number, address,
  or filename argument, and prints only the essential information leading
  up to and including the reference.  The -d option is not context
  specific, and only shows the data requested.

     -d dentry  given a hexadecimal dentry address, display its inode,
                super block, file type, and full pathname.
     -p inode   given a hexadecimal inode address, dump all of its pages
                that are in the page cache.
     -c         for each open file descriptor, prints a pointer to its
                inode, a pointer to the inode's i_mapping address_space
                structure, the number of pages of the inode that are in
                the page cache, the file type, and the pathname.
  -R reference  search for references to this file descriptor number,
                filename, dentry, inode, address_space, or file structure
                address.
           pid  a process PID.
         taskp  a hexadecimal task_struct pointer.

EXAMPLES
  Display the open files of the current context:

    crash> files
    PID: 720    TASK: c67f2000  CPU: 1   COMMAND: "innd"
    ROOT: /    CWD: /var/spool/news/articles
     FD    FILE     DENTRY    INODE    TYPE  PATH
      0  c6b9c740  c7cc45a0  c7c939e0  CHR   /dev/null
      1  c6b9c800  c537bb20  c54d0000  REG   /var/log/news/news
      2  c6df9600  c537b420  c5c36360  REG   /var/log/news/errlog
```

```
        3  c74182c0  c6ede260  c6da3d40  PIPE
        4  c6df9720  c696c620  c69398c0  SOCK
        5  c6b9cc20  c68e7000  c6938d80  SOCK
        6  c6b9c920  c7cc45a0  c7c939e0  CHR   /dev/null
        7  c6b9c680  c58fa5c0  c58a1200  REG   /var/lib/news/history
        8  c6df9f00  c6ede760  c6da3200  PIPE
        9  c6b9c6e0  c58fa140  c5929560  REG   /var/lib/news/history.dir
       10  c7fa9320  c7fab160  c7fafd40  CHR   /dev/console
       11  c6b9c7a0  c58fa5c0  c58a1200  REG   /var/lib/news/history
       12  c377ec60  c58fa5c0  c58a1200  REG   /var/lib/news/history
       13  c4528aa0  c58fa6c0  c52fbb00  REG   /var/lib/news/history.pag
       14  c6df9420  c68e7700  c6938360  SOCK
       15  c6df9360  c68e7780  c6938120  SOCK
       16  c6b9c0e0  c68e7800  c6772000  SOCK
       17  c6b9c200  c6b5f9c0  c6b5cea0  REG   /var/lib/news/active
       21  c6b9c080  c6ede760  c6da3200  PIPE

Display the files opened by the "crond" daemon, which is PID 462:

crash> files 462
  PID: 462    TASK: f7220000  CPU: 2   COMMAND: "crond"
  ROOT: /    CWD: /var/spool
   FD    FILE     DENTRY    INODE    TYPE  PATH
    0  f7534ae0  f7538de0  f7518dc0  CHR   /dev/console
    1  f7368f80  f72c7a40  f72f27e0  FIFO  pipe:/[1456]
    2  f74f3c80  f72c79c0  f72f2600  FIFO  pipe:/[1457]
    3  f7368b60  f72a5be0  f74300c0  REG   /var/run/crond.pid
    4  f7534360  f73408c0  f72c2840  REG   /var/log/cron
    7  f7368ce0  f72c7940  f72f2420  FIFO  pipe:/[1458]
    8  f7295de0  f72c7940  f72f2420  FIFO  pipe:/[1458]
   21  f74f36e0  f747cdc0  f747e840  CHR   /dev/null

The -R option is typically invoked from "foreach files".  This example
shows all tasks that have "/dev/pts/4" open:

  crash> foreach files -R pts/4
  PID: 18633  TASK: c310a000  CPU: 0   COMMAND: "crash"
  ROOT: /    CWD: /home/CVS_pool/crash
   FD    FILE     DENTRY    INODE    TYPE  PATH
    0  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4
    1  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4
    2  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4

  PID: 18664  TASK: c2392000  CPU: 1   COMMAND: "less"
  ROOT: /    CWD: /home/CVS_pool/crash
   FD    FILE     DENTRY    INODE    TYPE  PATH
    1  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4
    2  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4

  PID: 23162  TASK: c5088000  CPU: 1   COMMAND: "bash"
  ROOT: /    CWD: /home/CVS_pool/crash
   FD    FILE     DENTRY    INODE    TYPE  PATH
    0  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4
    1  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4
    2  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4
  255  c1412850  c2cb96d0  c2cad430  CHR   /dev/pts/4

  PID: 23159  TASK: c10fc000  CPU: 1   COMMAND: "xterm"
  ROOT: /    CWD: /homes/anderson/
   FD    FILE     DENTRY    INODE    TYPE  PATH
    5  c1560da0  c2cb96d0  c2cad430  CHR   /dev/pts/4

Display information about the dentry at address f745fd60:

  crash> files -d f745fd60
   DENTRY    INODE    SUPERBLK  TYPE  PATH
   f745fd60  f7284640  f73a3e00  REG   /var/spool/lpd/lpd.lock

For each open file, display the number of pages that are in the page cache:

  crash> files -c 1954
  PID: 1954   TASK: f7a28000  CPU: 1   COMMAND: "syslogd"
  ROOT: /    CWD: /
```

```
      FD   INODE    I_MAPPING  NRPAGES  TYPE  PATH
       0  cb3ae868   cb3ae910        0  SOCK  socket:/[4690]
       2  f2721c5c   f2721d04      461  REG   /var/log/messages
       3  cbda4884   cbda492c       47  REG   /var/log/secure
       4  e48092c0   e4809368       58  REG   /var/log/maillog
       5  f65192c0   f6519368       48  REG   /var/log/cron
       6  e4809e48   e4809ef0        0  REG   /var/log/spooler
       7  d9c43884   d9c4392c        0  REG   /var/log/boot.log
```

For the inode at address f59b90fc, display all of its pages that are in the page cache:

```
  crash> files -p f59b90fc
   INODE    NRPAGES
  f59b90fc        6

    PAGE    PHYSICAL   MAPPING   INDEX CNT FLAGS
  ca3353e0  39a9f000  f59b91ac        0  2 82c referenced,uptodate,lru,private
  ca22cb20  31659000  f59b91ac        1  2 82c referenced,uptodate,lru,private
  ca220160  3100b000  f59b91ac        2  2 82c referenced,uptodate,lru,private
  ca1ddde0  2eeef000  f59b91ac        3  2 82c referenced,uptodate,lru,private
  ca36b300  3b598000  f59b91ac        4  2 82c referenced,uptodate,lru,private
  ca202680  30134000  f59b91ac        5  2 82c referenced,uptodate,lru,private
```

NAME

mod - module information and loading of symbols and debugging data

SYNOPSIS

mod -s module [objfile] | -d module | -S [directory] [-D|-t|-r|-R|-o|-g]

DESCRIPTION

With no arguments, this command displays basic information of the currently installed modules, consisting of the module address, name, base address, size, the object file name (if known), and whether the module was compiled with CONFIG_KALLSYMS.

The arguments are concerned with with the loading or deleting of symbolic and debugging data from a module's object file. A modules's object file always contains symbolic data (symbol names and addresses), but contains debugging data only if the module was compiled with the -g CFLAG. In addition, the module may have compiled with CONFIG_KALLSYMS, which means that the module's symbolic data will have been loaded into the kernel's address space when it was installed. If the module was not compiled with CONFIG_KALLSYMS, then only the module's exported symbols will be loaded into the kernel's address space. Therefore, for the purpose of this command, it should noted that a kernel module may have been compiled in one of following manners:

1. If the module was built without CONFIG_KALLSYMS and without the -g CFLAG, then the loading of the module's additional non-exported symbols can be accomplished with this command.
2. If the module was built with CONFIG_KALLSYMS, but without the -g CFLAG, then there is no benefit in loading the symbols from the module object file, because all of the module's symbols will have been loaded into the kernel's address space when it was installed.
3. If the module was built with CONFIG_KALLSYMS and with the the -g CFLAG, then the loading of the module's debugging data can be accomplished with this command.
4. If the module was built without CONFIG_KALLSYMS but with the -g CFLAG, then the loading of the both module's symbolic and debugging data can be accomplished with this command.

-s module [objfile] Loads symbolic and debugging data from the object file for the module specified. If no objfile argument is appended, a search will be made for an object file consisting of the module name with a .o or .ko suffix, starting at the /lib/modules/<release> directory on the host system, or if not found there, starting at the directory containing the kernel namelist file. If an objfile argument is appended, then that file will be used.

```
            -d module  Deletes the symbolic and debugging data of the module
                       specified.
        -S [directory]  Load symbolic and debugging data from the object file
                       for all loaded modules.  For each module, a search
                       will be made for an object file consisting of the
                       module name with a .o or .ko suffix, starting at the
                       /lib/modules/<release> directory of the host system,
                       or if not found there, starting at the directory
                       containing the kernel namelist file.  If a directory
                       argument is appended, then the search will be restricted
                       to that directory.
                   -D  Deletes the symbolic and debugging data of all modules.
                   -t  Display the contents of the module's "taints" bitmask
                       if it is non-zero.  When possible, the "taints" bits
                       are translated to symbolic letters of the taint type;
                       otherwise the hexadecimal value is shown.  In older
                       kernels, the contents of the "license_gplok" field is
                       displayed in hexadecimal; the field may be either a
                       bitmask or a boolean, depending upon the kernel version.
                       The relevant kernel sources should be consulted for the
                       meaning of the letter(s) or hexadecimal bit value(s).
                       For modules that have a "gpgsig_ok" field that is zero
                       (unsigned), the notation "(U)" is shown.
                   -r  Passes the -readnow flag to the embedded gdb module,
                       which will override the two-stage strategy that it uses
                       for reading symbol tables from module object files.
                   -R  Reinitialize module data. All currently-loaded symbolic
                       and debugging data will be deleted, and the installed
                       module list will be updated (live system only).
                   -g  When used with -s or -S, add a module object's section
                       start and end addresses to its symbol list.
                   -o  Load module symbols with old mechanism.

  If the crash session was invoked with the "--mod <directory>" option, or
  a CRASH_MODULE_PATH environment variable exists, then /lib/modules/<release>
  will be overridden as the default directory tree that is searched for module
  object files.

  After symbolic and debugging data have been loaded, backtraces and text
  disassembly will be displayed appropriately.  Depending upon the processor
  architecture, data may also printed symbolically with the "p" command;
  at a minimum, the "rd" command may be used with module data symbols.

  If crash can recognize that the set of modules has changed while running a
  session on a live kernel, the module data will be reinitialized the next
  time this command is run; the -r option forces the reinitialization.

EXAMPLES
  Display the currently-installed modules:

    crash> mod
     MODULE   NAME              BASE      SIZE  OBJECT FILE
    f7e44c20  dm_mod           f7e34000   88568  (not loaded)
    f7e5a8a0  dm_log           f7e59000    8354  (not loaded)
    f7e66420  dm_region_hash   f7e65000    9708  (not loaded)
    f7e76b60  dm_mirror        f7e74000   12609  (not loaded)
    f7e8b8e0  ata_piix         f7e87000   20637  (not loaded)
    ...

  Display the currently-installed modules on a system where all modules were
  compiled with CONFIG_KALLSYMS:

    crash> mod
     MODULE   NAME              BASE      SIZE  OBJECT FILE
    f7e44c20  dm_mod           f7e34000   88568  (not loaded)  [CONFIG_KALLSYMS]
    f7e5a8a0  dm_log           f7e59000    8354  (not loaded)  [CONFIG_KALLSYMS]
    f7e66420  dm_region_hash   f7e65000    9708  (not loaded)  [CONFIG_KALLSYMS]
    f7e76b60  dm_mirror        f7e74000   12609  (not loaded)  [CONFIG_KALLSYMS]
    f7e8b8e0  ata_piix         f7e87000   20637  (not loaded)  [CONFIG_KALLSYMS]
    ...

  Load the symbolic and debugging data of all modules:
```

```
  crash> mod -S
   MODULE   NAME              BASE      SIZE  OBJECT FILE
  f7e44c20  dm_mod           f7e34000   88568
  /lib/modules/2.6.32/kernel/drivers/md/dm-mod.ko
  f7e5a8a0  dm_log           f7e59000    8354
  /lib/modules/2.6.32/kernel/drivers/md/dm-log.ko
  f7e66420  dm_region_hash   f7e65000    9708
  /lib/modules/2.6.32/kernel/drivers/md/dm-region-hash.ko
  f7e76b60  dm_mirror        f7e74000   12609
  /lib/modules/2.6.32/kernel/drivers/md/dm-mirror.ko
  f7e8b8e0  ata_piix         f7e87000   20637
  /lib/modules/2.6.32/kernel/drivers/ata/ata_piix.ko
  ...
```

Load the symbolic and debugging data of the dm_mod module from its known location:

```
  crash> mod -s dm_mod
   MODULE   NAME              BASE      SIZE  OBJECT FILE
  f7e44c20  dm_mod           f7e34000   88568
  /lib/modules/2.6.32/kernel/drivers/md/dm-mod.ko
```

Delete the current symbolic and debugging data of the dm_mod module, and then re-load it from a specified object file:

```
  crash> mod -d dm_mod
  crash> mod -s dm_mod /tmp/dm_mod.ko
   MODULE   NAME              BASE      SIZE  OBJECT FILE
  f7e44c20  dm_mod           f7e34000   88568  /tmp/dm-mod.ko
```

After installing a new kernel module on a live system, reinitialize the installed module list:

```
  crash> !modprobe soundcore
  crash> mod
  mod: NOTE: modules have changed on this system -- reinitializing
   MODULE   NAME              BASE      SIZE  OBJECT FILE
  f7e44c20  dm_mod           f7e34000   88568  (not loaded)
  f7e5a8a0  dm_log           f7e59000    8354  (not loaded)
  f7e62e40  soundcore        f7e62000    6390  (not loaded)
  f7e66420  dm_region_hash   f7e65000    9708  (not loaded)
  f7e76b60  dm_mirror        f7e74000   12609  (not loaded)
  f7e8b8e0  ata_piix         f7e87000   20637  (not loaded)
  ...
```

Display modules that are "tainted", where in this case where they are proprietary and unsigned:

```
  crash> mod -t
  NAME      TAINT
  vxspec    P(U)
  vxportal  P(U)
  fdd       P(U)
  vxfs      P(U)
  vxdmp     P(U)
  vxio      P(U)
  vxglm     P(U)
  vxgms     P(U)
  vxodm     P(U)
```

NAME

sbitmapq - sbitmap_queue struct contents

SYNOPSIS

```
sbitmapq [-s struct[.member[,member]] -a address [-p] [-v]] -[x|d] address
```

DESCRIPTION

The command dumps the contents of the sbitmap_queue structure and the used bits in the bitmap. Also, it shows the dump of a structure array associated with the sbitmap_queue.

The arguments are as follows:

```
 -s struct  name of a C-code structure, that is stored in an array
            associated with sbitmap_queue structure. Use the
            "struct.member" format in order to display a particular
            member of the structure. -s option requires -a option
-a address  address of a structure array associated with sbitmap_queue
            structure. The set bits in sbitmap are used for the index
            in an associated array.
         -p  associated with sbitmap_queue array contains the points of
            structure.
         -x  override default output format with hexadecimal format.
         -d  override default output format with decimal format.
         -v  By default, the sbitmap command shows only a used sbitmap
            index and a structure address in the associated array.
            This flag says to print a formatted display of the
            contents of a structure in an associated array. -v option
            requires of -s.

EXAMPLES

  All examples are shown on the base of Linux Target system with iSCSI
  transport.

  Display the common sbitmap information for target session:

    crash> struct -oh se_session 0xc0000000e118c760 | grep sbitmap_queue
      [c0000000e118c808] struct sbitmap_queue sess_tag_pool;
    crash>
    crash> sbitmapq c0000000e118c808
    depth = 136
    busy = 4
    cleared = 26
    bits_per_word = 32
    map_nr = 5
    alloc_hint = {74, 36, 123, 101}
    wake_batch = 8
    wake_index = 0
    ws_active = 0
    ws = {
            { .wait_cnt = 8, .wait = inactive },
            { .wait_cnt = 8, .wait = inactive },
            { .wait_cnt = 8, .wait = inactive },
            { .wait_cnt = 8, .wait = inactive },
            { .wait_cnt = 8, .wait = inactive },
            { .wait_cnt = 8, .wait = inactive },
            { .wait_cnt = 8, .wait = inactive },
            { .wait_cnt = 8, .wait = inactive },
    }
    round_robin = 0
    min_shallow_depth = 4294967295

    00000000: 0000 0000 0000 0000 0030 0000 0000 0000
    00000010: 00

  Display the addresses of structure are associated with
  sbitmap_queue (for iscsi it is 'iscsi_cmd' structure):

    crash> struct se_session 0xc0000000e118c760 | grep sess_cmd_map
      sess_cmd_map = 0xc0000000671c0000,
    crash>
    crash> sbitmapq -s iscsi_cmd -a 0xc0000000671c0000 c0000000e118c808
    76: 0xc0000000671d5600
    77: 0xc0000000671d5a80

  Dump of formatted content of structures:

    crash> sbitmapq -s iscsi_cmd -a 0xc0000000671c0000 -v c0000000e118c808
    76 (0xc0000000671d5600):
    struct iscsi_cmd {
      dataout_timer_flags = 0,
      dataout_timeout_retries = 0 '\000',
      error_recovery_count = 0 '\000',
      deferred_i_state = ISTATE_NO_STATE,
```

```
      i_state = ISTATE_SENT_STATUS,
      ...
      first_data_sg = 0xc0000000e306b080,
      first_data_sg_off = 0,
      kmapped_nents = 1,
      sense_reason = 0
    }
    77 (0xc0000000671d5a80):
    struct iscsi_cmd {
      dataout_timer_flags = 0,
      dataout_timeout_retries = 0 '\000',
      error_recovery_count = 0 '\000',
      deferred_i_state = ISTATE_NO_STATE,
      i_state = ISTATE_NEW_CMD,
      ...
      first_data_sg = 0x0,
      first_data_sg_off = 0,
      kmapped_nents = 0,
      sense_reason = 0
    }


NAME
  tree - display radix tree, XArray or red-black tree

SYNOPSIS
  tree [-t [radix|xarray|rbtree]] [-r offset] [-[s|S] struct[.member[,member]]]
       -[x|d] [-o offset] [-l] [-p] [-N] start

DESCRIPTION
  This command dumps the contents of a radix tree, an XAarray, or a red-black
  tree.  The arguments are as follows:

    -t type  The type of tree to dump; the type string can be one of
             "radix", "rbtree", or "xarray", or alternatively, "ra",
             "rb" or "x" are acceptable.  If not specified, rbtree is the
             default type.
  -r offset  If the "start" argument is the address of a data structure that
             contains an radix_tree_root, xarray or rb_root structure, then this
             is the offset to that structure member.  If the offset is non-zero,
             then this option is required.  The offset may be entered in either
             of two manners:
               1. In "structure.member" format.
               2. A number of bytes.
  -o offset  For red-black trees only, the offset of the rb_node within its
             containing data structure; if the offset is non-zero, then this
             option is required.  The offset may be entered in either of two
             manners:
               1. In "structure.member" format.
               2. A number of bytes.
             This option is not applicable to radix trees.
  -s struct  For each entry in a tree, format and print it as this type of data
             structure; use the "struct.member" format in order to display a
             particular member of the structure.  To display multiple members
             of a structure, use a comma-separated list of members.  If any
             structure member contains an embedded structure or is an array, the
             the output may be restricted to the embedded structure or an array
             element by expressing the member argument as "struct.member.member"
             or "struct.member[index]"; embedded member specifications may
             extend beyond one level deep by expressing the struct argument as
             "struct.member.member.member...".
  -S struct  Similar to -s, but instead of parsing gdb output, member values
             are read directly from memory, so the command works much faster
             for 1-, 2-, 4-, and 8-byte members.
         -l  For red-black trees, dump the tree sorted in linear order starting
             with the leftmost node and progressing to the right.  This option
             does not apply to radix trees.
         -p  Display the node's position information, showing the relationship
             between it and the root.  For red-black trees, a position that
             indicates "root/l/r" means that the node is the right child
             of the left child of the root node.  For radix trees and xarrays,
             the index, the height, and the slot index values are shown with
             respect to the root.
```

```
          -x  Override default output format with hexadecimal format.
          -d  Override default output format with decimal format.

   The meaning of the "start" argument, which can be expressed either in
   hexadecimal format or symbolically, depends upon whether the -N option
   is prepended:

       start  The address of a radix_tree_root, xarray or rb_root structure, or
              the address of a structure containing the radix_tree_root, xarray
              or rb_root structure; if the latter, then the "-r offset" option
              must be used if the member offset of the root structure is
              non-zero.

    -N start  The address of a radix_tree_node, xa_node or rb_node structure,
              bypassing the radix_tree_root, xarray, or rb_root that points
              to it.


EXAMPLES
   The vmap_area_root is a standalone rb_root structure.  Display the
   virtual addresses of each vmap_area in its red-black tree:

     crash> whatis vmap_area_root
     struct rb_root vmap_area_root;
     crash> tree -t rbtree -o vmap_area.rb_node vmap_area_root
     ffff880128c508c0
     ffff88012cb68140
     ffff88012c9afec0
     ffff88012d65c440
     ...

   Display the vmap_area's va_start and va_end members of each of
   the entries above expressing the vmap_area.rb_node offset as a
   number of bytes:

     crash> tree -t rbtree -o 24 vmap_area_root -s vmap_area.va_start,va_end
     ffff880128c508c0
       va_start = 0xffffc90014900000
       va_end = 0xffffc90014921000
     ffff88012cb68140
       va_start = 0xffffc900110c0000
       va_end = 0xffffc900110d1000
     ffff88012c9afec0
       va_start = 0xffffc90000640000
       va_end = 0xffffc90000642000
     ffff88012d65c440
       va_start = 0xffffc90000620000
       va_end = 0xffffc90000622000
     ...

   Alternatively, use the -N option with the rb_node address contained
   in the vmap_area_root structure:

     crash> p vmap_area_root
     vmap_area_root = $8 = {
       rb_node = 0xffff880128c508d8
     }
     crash> tree -t rbtree -o vmap_area.rb_node -N 0xffff880128c508d8
     ffff880128c508c0
     ffff88012cb68140
     ffff88012c9afec0
     ffff88012d65c440

   Display the virtual address of each vm_area_struct in the red-black
   tree that has its root inside an mm_struct located at ffff880128b5a300.
   The vm_area_struct.vm_rb rb_node member has an offset of 0x38 bytes:

     crash> tree -t rbtree -r mm_struct.mm_rb ffff880128b5a300 -o 0x38
     ffff88012a0de080
     ffff880123e3ac78
     ffff880123e3a700
     ffff88012b2837c8
     ...
```

```
  ffff880128c02ed0
  ffff8801292e7958
  ffff880123e3a318
  ffff880123e3ad40

Add the -p option to the command above to show position information:

  crash> tree -t rbtree -r mm_struct.mm_rb ffff880128b5a300 -o 0x38 -p
  ffff88012a0de080
    position: root
  ffff880123e3ac78
    position: root/l
  ffff880123e3a700
    position: root/l/l
  ffff88012b2837c8
    position: root/l/l/l
  ...
  ffff880128c02ed0
    position: root/r/r/l/r
  ffff8801292e7958
    position: root/r/r/l/r/r
  ffff880123e3a318
    position: root/r/r/r
  ffff880123e3ad40
    position: root/r/r/r/r

Given an mm_struct address of 0xffff880074b5be80, list the VMA tree in linear
order from the leftmost node progressing to the right using the -l option:

  crash> tree -ls vm_area_struct.vm_start -o vm_area_struct.vm_rb \
  -r mm_struct.mm_rb 0xffff880074b5be80 | paste - -
  ffff88001f2c50e0      vm_start = 0x400000
  ffff88001f2c5290      vm_start = 0xceb000
  ffff880074bfc6c0      vm_start = 0xcec000
  ffff88001f2c4bd0      vm_start = 0xd10000
  ffff880074bfc948      vm_start = 0x1fe9000
  ffff880036e54510      vm_start = 0x7ff6aa296000
  ffff88001f2c5bd8      vm_start = 0x7ff6aa298000
  ffff880036e54af8      vm_start = 0x7ff6aa497000
  ffff880036e54f30      vm_start = 0x7ff6aa498000
  ffff88000e06aa20      vm_start = 0x7ff6aa499000
  ffff88000e06b368      vm_start = 0x7ff6ab95f000
  ...
  ffff88001f2c5e60      vm_start = 0x7ff6bc1af000
  ffff88001f2c4ca8      vm_start = 0x7ff6bc1b6000
  ffff88001f2c5008      vm_start = 0x7ff6bc200000
  ffff88001f2c5d88      vm_start = 0x7ff6bc205000
  ffff880074bfd6c8      vm_start = 0x7ff6bc206000
  ffff88001f2c4288      vm_start = 0x7ff6bc207000
  ffff88001f2c4510      vm_start = 0x7ffc7a5fc000
  ffff88001f2c5b00      vm_start = 0x7ffc7a6d1000

Compared to the top/down root/leaves order:

  crash> tree -s vm_area_struct.vm_start -o vm_area_struct.vm_rb \
  -r mm_struct.mm_rb 0xffff880074b5be80 | paste - -
  ffff88001f2c5a28      vm_start = 0x7ff6bbbb9000
  ffff88001f2c55f0      vm_start = 0x7ff6bb252000
  ffff88000e06a360      vm_start = 0x7ff6ac6c3000
  ffff88001f2c4bd0      vm_start = 0xd10000
  ffff88001f2c5290      vm_start = 0xceb000
  ffff88001f2c50e0      vm_start = 0x400000
  ffff880074bfc6c0      vm_start = 0xcec000
  ffff88000e06b368      vm_start = 0x7ff6ab95f000
  ffff88001f2c5bd8      vm_start = 0x7ff6aa298000
  ffff880074bfc948      vm_start = 0x1fe9000
  ffff880036e54510      vm_start = 0x7ff6aa296000
  ffff880036e54f30      vm_start = 0x7ff6aa498000
  ffff880036e54af8      vm_start = 0x7ff6aa497000
  ffff88000e06aa20      vm_start = 0x7ff6aa499000
  ffff88000e06ae58      vm_start = 0x7ff6ac1df000
  ffff88000e06ba28      vm_start = 0x7ff6abefc000
  ffff88000e06a6c0      vm_start = 0x7ff6ac41b000
```

```
  ffff88001f2c4000      vm_start = 0x7ff6bac75000
  ffff88000e06bd88      vm_start = 0x7ff6b2d00000
  ffff88000e06b440      vm_start = 0x7ff6b28de000
  ...
  ffff880074bfd6c8      vm_start = 0x7ff6bc206000
  ffff88001f2c4510      vm_start = 0x7ffc7a5fc000
  ffff88001f2c5b00      vm_start = 0x7ffc7a6d1000
```

Display a list of the page structs in the radix tree of an address_space structure located at ffff88012d364de0:

```
  crash> tree -t radix -r address_space.page_tree ffff88012d364de0
  ffffea00040d12c0
  ffffea00040d9a60
  ffffea00040d9b08
  ffffea000407eda8
  ffffea0004084288
  ...
  ffffea000407bc70
  ffffea00040baf48
  ffffea0004043f48
  ffffea000407de58
```

Add the -p option to the command above to show position information:

```
  crash> tree -t radix -r address_space.page_tree ffff88012d364de0 -p
  ffffea00040d12c0
    index: 0  position: root/0/0
  ffffea00040d9a60
    index: 1  position: root/0/1
  ffffea00040d9b08
    index: 2  position: root/0/2
  ffffea000407eda8
    index: 3  position: root/0/3
  ffffea0004084288
    index: 4  position: root/0/4
  ...
  ffffea000407bc70
    index: 217  position: root/3/25
  ffffea00040baf48
    index: 218  position: root/3/26
  ffffea0004043f48
    index: 219  position: root/3/27
  ffffea000407de58
    index: 220  position: root/3/28
```

Alternatively, take the address of the radix_tree_node from the radix_tree_root structure in the address_space structure above, and display the tree with the -N option:

```
  crash> struct address_space.page_tree ffff88012d364de0
    page_tree = {
      height = 0x2,
      gfp_mask = 0x20,
      rnode = 0xffff8801238add71
    }
  crash> tree -t radix -N 0xffff8801238add71
  ffffea00040d12c0
  ffffea00040d9a60
  ffffea00040d9b08
  ffffea000407eda8
  ffffea0004084288
  ffffea00040843a0
  ...
```

Using the same radix tree as above, display the flags and _count members of each page struct in the list, and force the output format to be hexadecimal:

```
  crash> tree -t radix -N 0xffff8801238add71 -s page.flags,_count -x
  ffffea00040d12c0
    flags = 0x4000000002006c
    _count = {
```

```
        counter = 0x7
      }
    ffffea00040d9a60
      flags = 0x4000000002006c
      _count = {
        counter = 0x7
      }
    ffffea00040d9b08
      flags = 0x4000000002006c
      _count = {
        counter = 0x7
      }
    ffffea000407eda8
      flags = 0x4000000002006c
      _count = {
        counter = 0x7
      }
    ...
```

In more recent kernels, the XArray facility has replaced radix trees. Display a list of the page structs in the XArray of an address_space structure located at 0xffff94c235e76828, where the i_pages field is an embedded xarray structure:

```
  crash> tree -t xarray -r address_space.i_pages 0xffff94c235e76828
  ffffffcc005aa8380
  ffffffcc005cafa80
  ffffffcc005a79c80
  ffffffcc005ccad80
  ffffffcc005a72ec0
  ffffffcc005e27c00
  ffffffcc005ce3100
  ffffffcc005ff8dc0
  ffffffcc005c9a100
  ffffffcc005a49e40
  ffffffcc005c95a80
```

Add the -p option to the command above to show position information:

```
  crash> tree -t xarray -r address_space.i_pages 0xffff94c235e76828 -p
  ffffffcc005aa8380
    index: 90  position: root/1/26
  ffffffcc005cafa80
    index: 91  position: root/1/27
  ffffffcc005a79c80
    index: 92  position: root/1/28
  ffffffcc005ccad80
    index: 93  position: root/1/29
  ffffffcc005a72ec0
    index: 94  position: root/1/30
  ffffffcc005e27c00
    index: 95  position: root/1/31
  ffffffcc005ce3100
    index: 96  position: root/1/32
  ffffffcc005ff8dc0
    index: 97  position: root/1/33
  ffffffcc005c9a100
    index: 98  position: root/1/34
  ffffffcc005a49e40
    index: 99  position: root/1/35
  ffffffcc005c95a80
    index: 100  position: root/1/36
```

Alternatively, take the value found in the xa_head field from the xarray structure, and display the tree with the -N option:

```
  crash> address_space.i_pages 0xffff94c235e76828
    i_pages = {
      ... [ xa_lock field not shown ] ...
      xa_flags = 1,
      xa_head = 0xffff94c23c1566ca
    }
  crash> tree -t x -N 0xffff94c23c1566ca
```

```
  ffffcc005aa8380
  ffffcc005cafa80
  ffffcc005a79c80
  ffffcc005ccad80
  ffffcc005a72ec0
  ffffcc005e27c00
  ffffcc005ce3100
  ffffcc005ff8dc0
  ffffcc005c9a100
  ffffcc005a49e40
  ffffcc005c95a80

Using the same xarray command as above, display the flags and _refcount
members of each page struct in the list, and force the output format
to be hexadecimal:

  crash> tree -t x -N 0xffff94c23c1566ca -s page.flags,_refcount -x
  ffffcc005aa8380
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005cafa80
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005a79c80
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005ccad80
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005a72ec0
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005e27c00
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005ce3100
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005ff8dc0
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005c9a100
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005a49e40
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
  ffffcc005c95a80
    flags = 0x57ffffc0000014
    _refcount = {
      counter = 0x1
    }
```

```
NAME
  alias - command aliases

SYNOPSIS
  alias [alias] [command string]

DESCRIPTION
  This command creates an alias for a given command string.  If no arguments
  are entered, the current list of aliases are displayed.  If one argument is
  entered, the command string for that alias, if any, is displayed.

           alias  the single word to be used as an alias
  command string  the word(s) that will be substituted for the alias

  Aliases may be created in four manners:

    1. entering the alias in $HOME/.crashrc.
    2. entering the alias in .crashrc in the current directory.
    3. executing an input file containing the alias command.
    4. during runtime with this command.

  During initialization, $HOME/.crashrc is read first, followed by the
  .crashrc file in the current directory.  Aliases in the .crashrc file
  in the current directory override those in $HOME/.crashrc.  Aliases
  entered with this command or by runtime input file override those
  defined in either .crashrc file.  Aliases may be deleted by entering an
  empty string for the second argument.  If redirection characters are to
  be part of the command string, the command string must be enclosed by
  quotation marks.

  Note that there are a number of helpful built-in aliases -- see the
  first example below.

EXAMPLES
  Display the currently-defined aliases, which in this example, only
  consist of the built-in aliases:

    crash> alias
    ORIGIN   ALIAS    COMMAND
    builtin  man      help
    builtin  ?        help
    builtin  quit     q
    builtin  sf       set scroll off
    builtin  sn       set scroll on
    builtin  hex      set radix 16
    builtin  dec      set radix 10
    builtin  g        gdb
    builtin  px       p -x
    builtin  pd       p -d
    builtin  for      foreach
    builtin  size     *
    builtin  dmesg    log
    builtin  lsmod    mod
    builtin  last     ps -l

  Create a new alias to be added to the list:

    crash> alias kp kmem -p
    ORIGIN   ALIAS    COMMAND
    runtime  kp       kmem -p

  Create an alias with redirection characters:

    crash> alias ksd "kmem -p | grep slab | grep DMA"
    ORIGIN   ALIAS    COMMAND
    runtime  ksd      kmem -p | grep slab | grep DMA

  Remove an alias:

    crash> alias kp ""
    alias deleted: kp
```

```
NAME
  foreach - display command data for multiple tasks in the system

SYNOPSIS
  foreach [[pid | taskp | name | state | [kernel | user | gleader]] ...]
          command [flag] [argument]

DESCRIPTION
  This command allows for an examination of various kernel data associated
  with any, or all, tasks in the system, without having to set the context
  to each targeted task.

      pid  perform the command(s) on this PID.
    taskp  perform the command(s) on task referenced by this hexadecimal
           task_struct pointer.
     name  perform the command(s) on all tasks with this name.  If the
           task name can be confused with a foreach command name, then
           precede the name string with a "\".  If the name string is
           enclosed within "'" characters, then the encompassed string
           must be a POSIX extended regular expression that will be used
           to match task names.
     user  perform the command(s) on all user (non-kernel) threads.
  gleader  perform the command(s) on all user (non-kernel) thread group leaders.
   kernel  perform the command(s) on all kernel threads.
   active  perform the command(s) on the active thread on each CPU.
    state  perform the command(s) on all tasks in the specified state, which
           may be one of: RU, IN, UN, ST, ZO, TR, SW, DE, WA, PA, ID or NE.

  If none of the task-identifying arguments above are entered, the command
  will be performed on all tasks.

  command  select one or more of the following commands to be run on the tasks
           selected, or on all tasks:

              bt  run the "bt" command  (optional flags: -r -t -l -e -R -f -F
                  -o -s -x -d)
              vm  run the "vm" command  (optional flags: -p -v -m -R -d -x)
            task  run the "task" command  (optional flags: -R -d -x)
           files  run the "files" command  (optional flag: -c -R)
             net  run the "net" command  (optional flags: -s -S -R -d -x)
             set  run the "set" command
              ps  run the "ps" command  (optional flags: -G -s -p -c -t -l -a
                  -g -r -y)
             sig  run the "sig" command (optional flag: -g)
            vtop  run the "vtop" command  (optional flags: -c -u -k)

     flag  Pass this optional flag to the command selected.
 argument  Pass this argument to the command selected.

  A header containing the PID, task address, cpu and command name will be
  pre-pended before the command output for each selected task.  Consult the
  help page of each of the command types above for details.

EXAMPLES
  Display the stack traces for all tasks:

    crash> foreach bt
    PID: 4752   TASK: c7680000  CPU: 1   COMMAND: "xterm"
     #0 [c7681edc] schedule at c01135f6
        (void)
     #1 [c7681f34] schedule_timeout at c01131ff
        (24)
     #2 [c7681f64] do_select at c0132838
        (5, c7681fa4, c7681fa0)
     #3 [c7681fbc] sys_select at c0132dad
        (5, 8070300, 8070380, 0, 0)
     #4 [bffffb0c] system_call at c0109944
        EAX: 0000008e  EBX: 00000005  ECX: 08070300  EDX: 08070380
        DS:  002b      ESI: 00000000  ES:  002b      EDI: 00000000
        SS:  002b      ESP: bffffadc  EBP: bffffb0c
        CS:  0023      EIP: 402259ee  ERR: 0000008e  EFLAGS: 00000246
```

```
    PID: 557    TASK: c5600000  CPU: 0   COMMAND: "nfsd"
     #0 [c5601f38] schedule at c01135f6
        (void)
     #1 [c5601f90] schedule_timeout at c01131ff
        (c5600000)
     #2 [c5601fb8] svc_recv at c805363a
        (c0096f40, c5602800, 7fffffff, 100, c65c9f1c)
     #3 [c5601fec] (nfsd module) at c806e303
        (c5602800, c5602800, c0096f40, 6c6e0002, 50)
     #4 [c65c9f24] kernel_thread at c010834f
        (0, 0, ext2_file_inode_operations)

    PID: 824    TASK: c7c84000  CPU: 0   COMMAND: "mingetty"
    ...
```

Display the task_struct structure for each "bash" command:

```
    crash> foreach bash task
    ...
```

Display the open files for all tasks:

```
    crash> foreach files
    ...
```

Display the state of tasks whose name contains a match to "event.*":

```
    crash> foreach 'event.*' task -R state
    PID: 99     TASK: ffff8804750d5500  CPU: 0   COMMAND: "events/0"
      state = 1,

    PID: 100    TASK: ffff8804750d4ac0  CPU: 1   COMMAND: "events/1"
      state = 1,

    PID: 101    TASK: ffff8804750d4080  CPU: 2   COMMAND: "events/2"
      state = 1,
    ...
```

Display the stack traces for all blocked (TASK_UNINTERRUPTIBLE) tasks:

```
    crash> foreach UN bt
    PID: 428    TASK: ffff880036b6c560  CPU: 1   COMMAND: "jbd2/dm-1-8"
     #0 [ffff880035779a70] __schedule at ffffffff815df272
     #1 [ffff880035779b08] schedule at ffffffff815dfacf
     #2 [ffff880035779b18] io_schedule at ffffffff815dfb7f
     #3 [ffff880035779b38] sleep_on_page at ffffffff81119a4e
     #4 [ffff880035779b48] __wait_on_bit at ffffffff815e039f
     #5 [ffff880035779b98] wait_on_page_bit at ffffffff81119bb8
     #6 [ffff880035779be8] filemap_fdatawait_range at ffffffff81119ccc
     #7 [ffff880035779cd8] filemap_fdatawait at ffffffff81119d8b
     #8 [ffff880035779ce8] jbd2_journal_commit_transaction at ffffffff8123a99c
     #9 [ffff880035779e58] kjournald2 at ffffffff8123ee7b
    #10 [ffff880035779ee8] kthread at ffffffff8108fb9c
    #11 [ffff880035779f48] kernel_thread_helper at ffffffff815ebaf4
    ...
```

```
NAME
  mount - mounted filesystem data

SYNOPSIS
  mount [-f][-i] [-n pid|task] [mount|vfsmount|superblock|dev|dir|dentry|inode]

DESCRIPTION
  This command displays basic information about the currently-mounted
  filesystems.  The per-filesystem dirty inode list or list of open
  files for the filesystem may also be displayed.

     -f  dump dentries and inodes for open files in each filesystem; only
         supported on kernels prior to Linux 3.13.
     -i  dump all dirty inodes associated with each filesystem; only
         supported on kernels prior to Linux 2.6.32.
```

```
   For kernels supporting namespaces, the -n option may be used to
   display the mounted filesystems with respect to the namespace of a
   specified task:

      -n pid   a process PID.
      -n task  a hexadecimal task_struct pointer.

   Specific filesystems may be selected using the following forms:

     vfsmount  hexadecimal address of a filesystem vfsmount structure.
        mount  hexadecimal address of a filesystem mount structure (Linux 3.3
               and later).
   superblock  hexadecimal address of a filesystem super_block structure.
          dev  device name of a filesystem.
          dir  directory where a filesystem is mounted.
       dentry  hexadecimal address of an open dentry of a filesystem.
        inode  hexadecimal address of an open inode of a filesystem.

   The first column of the command output displays the filesystem's vfsmount
   structure address for kernels prior to Linux 3.3.  For Linux 3.3 and later
   kernels, the first column displays the filesystem's mount structure address,
   which contains an embedded vfsmount structure.

EXAMPLES
   Display mounted filesystem data:

     crash> mount
     VFSMOUNT SUPERBLK TYPE   DEVNAME   DIRNAME
     c0089ea0 c0088a00 ext2   /dev/root /
     c0089cf0 c0088c00 proc   /proc     /proc
     c0089e10 c0088800 ext2   /dev/sda5 /boot
     c0089d80 c0088600 ext2   /dev/sda6 /usr
     c0089f30 c0088400 devpts none      /dev/pts
     c3f4b010 c0088200 ext2   /dev/sda1 /home

   On Linux 3.3 and later kernels, the filesystem's mount structure address
   is shown:

     crash> mount
          MOUNT           SUPERBLK     TYPE   DEVNAME   DIRNAME
     ffff880212fb8200 ffff880212fc0800 rootfs rootfs    /
     ffff88020ffbea00 ffff880212fc2000 proc   proc      proc
     ffff880211db7f00 ffff88020e01a800 sysfs  sysfs     /sys
     ffff88020ffe1300 ffff880212a40000 devtmpfs devtmpfs /dev
     ffff88020ff15000 ffff880212bbc800 devpts devpts    /dev/pts
     ffff88020e542800 ffff88020e62b800 tmpfs  tmpfs     /dev/shm
     ...

   Display the open files associated with each mounted filesystem:

     crash> mount -f
     VFSMOUNT SUPERBLK TYPE   DEVNAME   DIRNAME
     c7fb2b80 c7fb3200 ext2   /dev/root /
     OPEN FILES:
      DENTRY    INODE    TYPE  PATH
     c6d02200  c6d0f7a0  REG   usr/X11R6/lib/libX11.so.6.1
     c6d02100  c6d0f9e0  REG   usr/X11R6/lib/libXext.so.6.3
     c6d02000  c6d0fc20  REG   usr/X11R6/lib/libICE.so.6.3
     c6d02680  c6d0f320  REG   usr/X11R6/bin/xfs
     c7106580  c70c5440  CHR   dev/psaux
     ...

   Display the dirty inodes associated with each mounted filesystem:

     crash> mount -i
     VFSMOUNT SUPERBLK TYPE   DEVNAME   DIRNAME
     c0089ea0 c0088a00 ext2   /dev/root /
     DIRTY INODES
     c7ad4008
     c2233438
     c72c4008
     c7d6b548
```

```
      c3af1a98
      c7d6b768
      c3c4e228
      ...

  Display the mounted filesystem containing inode c5000aa8:

    crash> mount c5000aa8
    VFSMOUNT SUPERBLK TYPE   DEVNAME   DIRNAME
    c0089f30 c0088600 ext2   /dev/sda6 /usr

  Display the mounted filesystem containing inode ffff8801f4245e40:

    crash> mount ffff8801f4245e40
         MOUNT           SUPERBLK     TYPE   DEVNAME  DIRNAME
    ffff88020ffbea00 ffff880212fc2000 proc   proc     /proc



NAME
  search - search memory

SYNOPSIS
  search [-s start] [ -[kKV] | -u | -p | -t | -T ] [-e end | -l length] [-m mask]
         [-x count] -[cwh] [value | (expression) | symbol | string] ...

DESCRIPTION
  This command searches for a given value within a range of user virtual, kernel
  virtual, or physical memory space.  If no end nor length value is entered,
  then the search stops at the end of user virtual, kernel virtual, or physical
  address space, whichever is appropriate.

  An optional mask value may be entered to mask off "don't care" bits.

    -s start  Start the search at this hexadecimal user or kernel virtual
              address, physical address, or kernel symbol.  The start address
              must be appropriate for the memory type specified; if no memory
              type is specified, the default is kernel virtual address space.
          -k  If no start address is specified, start the search at the base
              of kernel virtual address space.  This option is the default.
          -K  Same as -k, except that mapped kernel virtual memory that was
              allocated by vmalloc(), module memory, or virtual mem_map regions
              will not be searched.
          -V  Same as -k, except that unity-mapped kernel virtual memory and
              mapped kernel-text/static-data (x86_64 and ia64) will not be
              searched.
          -u  If no start address is specified, start the search at the base
              of the current context's user virtual address space.  If a start
              address is specified, then this option specifies that the start
              address is a user virtual address.
          -p  If no start address is specified, start the search at the base
              of physical address space.  If a start address is specified,
              then this option specifies that the start address is a physical
              address.
          -t  Search only the kernel stack pages of every task.  If one or more
              matches are found in a task's kernel stack, precede the output
              with a task-identifying header.
          -T  Same as -t, except only the active task(s) are considered.
      -e end  Stop the search at this hexadecimal user or kernel virtual
              address, kernel symbol, or physical address.  The end address
              must be appropriate for the memory type specified.
   -l length  Length in bytes of address range to search.
     -m mask  Ignore the bits that are set in the hexadecimal mask value.
          -c  Search for character string values instead of unsigned longs.  If
              the string contains any space(s), it must be encompassed by double
              quotes.
          -w  Search for unsigned hexadecimal ints instead of unsigned longs.
              This is only meaningful on 64-bit systems in order to search both
              the upper and lower 32-bits of each 64-bit long for the value.
          -h  Search for unsigned hexadecimal shorts instead of unsigned longs.
    -x count  Display the memory contents before and after any found value.  The
              before and after memory context will consist of "count" memory
              items of the same size as the "value" argument.  This option is
```

```
                not applicable with the -c option.
         value  Search for this hexadecimal long, unless modified by the -c, -w,
                or -h options.
  (expression)  Search for the value of this expression; the expression value must
                not overflow the designated size when -h or -w are used; not
                applicable when used with the -c option.
        symbol  Search for this symbol value; the symbol value must not overflow
                the designated size when -h or -w are used; not applicable when
                used with the -c option.
        string  Search for character string values; if the string contains any
                space(s), it must be encompassed by double quotes; only applicable
                with the -c option.

  If -k, -K, -V, -u, -p or -t are not used, then the search defaults to kernel
  virtual address space.  The starting address must be long-word aligned.
  Address ranges that start in user space and end in kernel space are not
  accepted.

EXAMPLES
  Search the current context's address space for all instances of 0xdeadbeef:

    crash> search -u deadbeef
    81aba5c: deadbeef
    81abaa8: deadbeef
    bfffc698: deadbeef
    bffff390: deadbeef

  Search all kernel memory above the kernel text space for all instances
  of 0xabcd occurring in the lower 16-bits of each 32-bit word:

    crash> search -s _etext -m ffff0000 abcd
    c071481c: abcd
    c0c2b0fc: 804abcd
    c0cf5e74: 7489abcd
    c17c0b44: c012abcd
    c1dac730: 3dbeabcd
    c226d0e8: ffffabcd
    c23ed5dc: abcd
    c3022544: 3dbeabcd
    c3069b58: 3dbeabcd
    c3e86e84: aabcd
    c3e88ed0: aabcd
    c3e8ee5c: aabcd
    c3e9df50: aabcd
    c3e9e930: aabcd
    c440a778: 804abcd
    c486eb44: 3dbeabcd
    c578f0fc: 804abcd
    c6394f90: 8ababcd
    c65219f0: 3abcd
    c661399c: abcd
    c68514ac: 8abcd
    c7e036bc: 3dbeabcd
    c7e12568: 5abcd
    c7e1256c: 5abcd

  Search the 4K page at c532c000 for all instances of 0xffffffff:

    crash> search -s c532c000 -l 4096 ffffffff
    c532c33c: ffffffff
    c532c3fc: ffffffff

  Search the static kernel data area for all instances of c2d400eb:

    crash> search -s _etext -e _edata c2d400eb
    c022b550: c2d400eb
    c022b590: c2d400eb
    c022b670: c2d400eb
    c022b6e0: c2d400eb
    c022b7b0: c2d400eb
    c022b7e0: c2d400eb
    c022b8b0: c2d400eb
```

Search physical memory for all instances of 0xbabe occurring in the
upper 16 bits of each 32-bit word:

```
crash> search -p babe0000 -m ffff
2a1dc4: babe671e
2b6928: babe3de1
2f99ac: babe0d54
31843c: babe70b9
3ba920: babeb5d7
413ce4: babe7540
482747c: babe2600
48579a4: babe2600
4864a68: babe2600
...
```

Search physical memory for all instances of 0xbabe occurring in the
upper 16 bits of each 32-bit word on a 64-bit system:

```
crash> search -p babe0000 -m ffff -w
102e248: babe1174
11d2f90: babe813d
122d3ad70: babe6b27
124d8cd30: babe3dc8
124d8eefc: babef981
124d8f060: babe3dc8
124d8f17c: babefc81
...
```

Search kernel memory for all instances of 32-bit value 0xbabe1174
on a 64-bit system:

```
crash> search -k -w babe1174
ffff88000102e248: babe1174
ffffffff8102e248: babe1174
```

Search kernel memory for two strings:

```
crash> search -k -c "can't allocate memory" "Failure to"
ffff8800013ddec1: can't allocate memory for key lists..<3>%s %s: error con
ffff8801258be748: Failure to install fence: %d..<3>[drm:%s] *ERROR* Failed
ffff880125f07ec9: can't allocate memory..<3>ACPI: Invalid data..Too many d
ffffffff813ddec1: can't allocate memory for key lists..<3>%s %s: error con
```

Search the kernel stacks of all tasks for those that contain the inode
address ffff81002c0a3050:

```
crash> search -t ffff81002c0a3050
PID: 4876   TASK: ffff81003e9f5860  CPU: 7   COMMAND: "automount"
ffff8100288fbe98: ffff81002c0a3050

PID: 4880   TASK: ffff81003ce967a0  CPU: 0   COMMAND: "automount"
ffff81002c0fbdd8: ffff81002c0a3050
ffff81002c0fbe78: ffff81002c0a3050
```

When a kernel symbol or an (expression) is used an argument, both the
resultant value and the input string are displayed:

```
crash> search anon_inode_inode (__down_interruptible+191)
ffff81000222a728: ffffffff80493d60 (anon_inode_inode)
ffff810005a1e918: ffffffff800649d6 (__down_interruptible+191)
ffff810005a1e9d0: ffffffff800649d6 (__down_interruptible+191)
ffff810005a1eb48: ffffffff800649d6 (__down_interruptible+191)
ffff81000b409c60: ffffffff80493d60 (anon_inode_inode)
ffff81000c155b98: ffffffff80493d60 (anon_inode_inode)
ffff8100194fac70: ffffffff80493d60 (anon_inode_inode)
ffff81001daa1008: ffffffff80493d60 (anon_inode_inode)
ffff810028b95830: ffffffff800649d6 (__down_interruptible+191)
ffff81002cea0c70: ffffffff80493d60 (anon_inode_inode)
ffff810031327268: ffffffff80493d60 (anon_inode_inode)
ffff810031327270: ffffffff800649d6 (__down_interruptible+191)
ffff810034b1ccd0: ffffffff800649d6 (__down_interruptible+191)
ffff8100399565a8: ffffffff80493d60 (anon_inode_inode)
ffff81003a278cd0: ffffffff800649d6 (__down_interruptible+191)
```

```
      ffff81003cc23e08: ffffffff800649d6 (__down_interruptible+191)


NAME
  union - union contents

SYNOPSIS
  union union_name[.member[,member]] [-o][-l offset][-rfuxdp]
         [address | symbol][:cpuspec] [count | -c count]

DESCRIPTION
  This command displays either a union definition, or a formatted display
  of the contents of a union at a specified address.  When no address is
  specified, the union definition is shown along with the union size.
  A union member may be appended to the structure name in order to limit
  the scope of the data displayed to that particular member; when no address
  is specified, the member's offset (always 0) and definition are shown.

     union_name  name of a C-code union used by the kernel.
        .member  name of a union member; to display multiple members of a
                 union, use a comma-separated list of members.  If any member
                 contains an embedded structure, or the member is an array, the
                 output may be restricted to just the embedded structure or an
                 array element by expressing the argument as "member.member"
                 or "member[index]"; embedded member specifications may extend
                 beyond one level deep, by expressing the member argument as
                 "member.member.member...".
             -o  show member offsets when displaying union definitions; the
                 offset is always 0 unless used with an address or symbol
                 argument, in which case each member will be preceded by its
                 virtual address.
      -l offset  if the address argument is a pointer to a list_head structure
                 that is embedded in the target union structure, the offset
                 to the list_head member may be entered in either of the
                 following manners:
                   1. in "structure.member" format.
                   2. a number of bytes.
             -r  raw dump of union data.
             -f  address argument is a dumpfile offset.
             -x  override default output format with hexadecimal format.
             -d  override default output format with decimal format.
             -p  if a union member is a pointer value, show the member's
                 data type on the output line; and on the subsequent line(s),
                 dereference the pointer, display the pointer target's symbol
                 value in brackets if appropriate, and if possible, display the
                 target data; requires an address argument.
             -u  address argument is a user virtual address in the current
                 context.
       address  hexadecimal address of a union; if the address points
                 to an embedded list_head structure contained within the
                 target union structure, then the "-l" option must be used.
        symbol  symbolic reference to the address of a union.
      :cpuspec  CPU specification for a per-cpu address or symbol:
                  :             CPU of the currently selected task.
                  :a[ll]        all CPUs.
                  :#[-#][,...]  CPU list(s), e.g. "1,3,5", "1-3",
                                or "1,3,5-7,10".
         count  count of unions to dump from an array of unions; if used,
                 this must be the last argument entered.
      -c count  "-c" is only required if "count" is not the last argument
                 entered or if a negative number is entered; if a negative
                 value is entered, the (positive) "count" structures that
                 lead up to and include the target structure will be displayed.

  Union data, sizes, and member offsets are shown in the current output radix
  unless the -x or -d option is specified.

  Please note that in the vast majority of cases, the "union" command
  name may be dropped; if the union name does not conflict with any crash
  or gdb command name, then the "union_name[.member]" argument will be
  recognized as a union name, and this command automatically executed.
  See the NOTE below.
```

```
EXAMPLES

  Display the bdflush_param union definition, and then an instance of it:

    crash> union bdflush_param
    union bdflush_param {
        struct {
            int nfract;
            int ndirty;
            int nrefill;
            int nref_dirt;
            int dummy1;
            int age_buffer;
            int age_super;
            int dummy2;
            int dummy3;
        } b_un;
        unsigned int data[9];
    }

    SIZE: 36  (0x24)

    crash> union bdflush_param bdf_prm
    union bdflush_param {
      b_un = {
        nfract = 40,
        ndirty = 500,
        nrefill = 64,
        nref_dirt = 256,
        dummy1 = 15,
        age_buffer = 3000,
        age_super = 500,
        dummy2 = 1884,
        dummy3 = 2
      },
      data = {40, 500, 64, 256, 15, 3000, 500, 1884, 2}
    }

NOTE
  If the union name does not conflict with any crash command name, the
  "union" command may be dropped.  Accordingly, the examples above could
  also have been accomplished like so:

    crash> bdflush_param
    crash> bdflush_param bdf_prm

  Lastly, the short-cut "*" (pointer-to) command may also be used to negate
  the need to enter the "union" command name (enter "help *" for details).


NAME
  ascii - translate a hexadecimal string to ASCII

SYNOPSIS
  ascii value ...

DESCRIPTION
  Translates 32-bit or 64-bit hexadecimal values to ASCII.  If no argument
  is entered, an ASCII chart is displayed.

EXAMPLES
  Translate the hexadecimal value of 0x62696c2f7273752f to ASCII:

    crash> ascii 62696c2f7273752f
    62696c2f7273752f: /usr/lib

  Display an ASCII chart:

    crash> ascii

          0    1   2   3   4   5   6   7
        +-------------------------------
      0 | NUL DLE  SP  0   @   P   '   p
```

```
       1 | SOH DC1  !   1   A   Q   a   q
       2 | STX DC2  "   2   B   R   b   r
       3 | ETX DC3  #   3   C   S   c   s
       4 | EOT DC4  $   4   D   T   d   t
       5 | ENQ NAK  %   5   E   U   e   u
       6 | ACK SYN  &   6   F   V   f   v
       7 | BEL ETB  `   7   G   W   g   w
       8 |  BS CAN  (   8   H   X   h   x
       9 |  HT  EM  )   9   I   Y   i   y
       A |  LF SUB  *   :   J   Z   j   z
       B |  VT ESC  +   ;   K   [   k   {
       C |  FF  FS  ,   <   L   \   l   |
       D |  CR  GS  _   =   M   ]   m   }
       E |  SO  RS  .   >   N   ^   n   ~
       F |  SI  US  /   ?   O   -   o  DEL


NAME
  fuser - file users

SYNOPSIS
  fuser [pathname | inode]

DESCRIPTION
  This command displays the tasks using specified files or sockets.
  Tasks will be listed that reference the file as the current working
  directory, root directory, an open file descriptor, or that mmap the
  file.  If the file is held open in the kernel by the lockd server on
  behalf of a client discretionary file lock, the client hostname is
  listed.

    pathname  the full pathname of the file.
    inode     the hexadecimal inode address for the file.

EXAMPLES
  Display the tasks using file /usr/lib/libkfm.so.2.0.0

    crash> fuser /usr/lib/libkfm.so.2.0.0
     PID    TASK    COMM            USAGE
     779  c5e82000  "kwm"           mmap
     808  c5a8e000  "krootwm"       mmap
     806  c5b42000  "kfm"           mmap
     809  c5dde000  "kpanel"        mmap


NAME
  net - network command

SYNOPSIS
  net [[-s | -S] [-xd] [-R ref] [pid | task]] [-a] [ -n [pid | task]] [-N addr]

DESCRIPTION
  Displays various network related data.

  If no arguments are entered, the list of network devices, names and IP
  addresses are displayed.  For kernels supporting namespaces, the -n option
  may be used to display the list of network devices with respect to the
  network namespace of a current context or a task specified by pid or task:

        -n  the namespace of the current context.
    -n pid  a process PID.
   -n task  a hexadecimal task_struct pointer.

  The -s and -S options display data with respect to the current context, but
  may be appended with an argument to show the socket data with respect
  to a specified task:

        -s  display open network socket/sock addresses, their family and type,
            and for INET and INET6 families, their source and destination
            addresses and ports.
    -s pid  same as above, for task with process PID pid.
   -s task  same as above, for task with hexadecimal task_struct pointer task.
```

```
          -S  displays open network socket/sock addresses followed by a dump
              of both data structures.
       -S pid  same as above, with respect to process PID.
      -S task  same as above, with respect to hexadecimal task_struct pointer.

    The -R option, typically invoked from "foreach net", and in conjunction
    with the -s or -S options, searches for references to a socket address,
    sock address, or a file descriptor; if found, only the referenced fd, socket
    or sock data will be displayed:

       -R ref  socket or sock address, or file descriptor.

    Other options:

          -a  display the ARP cache.
      -N addr  translates an IPv4 address expressed as a decimal or hexadecimal
               value into a standard numbers-and-dots notation.
          -x  override default output format with hexadecimal format.
          -d  override default output format with decimal format.

EXAMPLES
    Display the system's network device list:

      crash> net
         NET_DEVICE     NAME   IP ADDRESS(ES)
      ffff8803741c0000  lo     127.0.0.1
      fff88037059c0000  eth0   10.226.229.141
      ffff8803705c0000  eth1   10.226.228.250
      ffff880374ad6000  usb0   169.254.95.120

    Display the network device list with respect to the network namespace
    of PID 2618:

      crash> net -n 2618
         NET_DEVICE     NAME   IP ADDRESS(ES)
      ffff880456ee7020  lo     127.0.0.1
      ffff8804516a1020  eth0   10.1.9.223

    Dump the ARP cache:

      crash> net -a
      NEIGHBOUR      IP ADDRESS     HW TYPE   HW ADDRESS         DEVICE  STATE
      f38d1b00       10.16.64.14    ETHER     00:16:3e:4b:a5:4a  eth1    STALE
      f38d1080       0.0.0.0        UNKNOWN   00 00 00 00 00 00  lo      NOARP
      f38d1bc0       10.16.71.254   ETHER     00:00:0c:07:ac:00  eth1    REACHABLE
      f38d1200       10.16.64.21    ETHER     00:16:3e:51:d8:09  eth1    REACHABLE

    Display the sockets for PID 2517, using both -s and -S output formats:

      crash> net -s 2517
      PID: 2517   TASK: c1598000  CPU: 1   COMMAND: "rlogin"
      FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
       3  c57375dc  c1ff1850  INET:STREAM      10.1.8.20-1023      10.1.16.62-513

      crash> net -S 2517
      PID: 2517   TASK: c1598000  CPU: 1   COMMAND: "rlogin"
      FD   SOCKET     SOCK
       3  c57375dc  c1ff1850

      struct socket {
        state = SS_CONNECTED,
        flags = 131072,
        ops = 0xc023f820,
        inode = 0xc5737540,
        fasync_list = 0x0,
        file = 0xc58892b0,
        sk = 0xc1ff1850,
        wait = 0xc14d9ed4,
        type = 1,
        passcred = 0 '\000',
        tli = 0 '\000'
      }
      struct sock {
```

```
        sklist_next = 0xc1ff12f0,
        sklist_prev = 0xc216bc00,
        bind_next = 0x0,
        bind_pprev = 0xc0918448,
        daddr = 1041236234,
        rcv_saddr = 336068874,
        dport = 258,
        num = 1023,
        bound_dev_if = 0,
        next = 0x0,
        pprev = 0xc0286dd4,
        state = 1 '\001',
        zapped = 0 '\000',
        sport = 65283,
        family = 2,
        reuse = 0 '\000',
        ...
```

Translate the rcv_saddr from above into dotted-decimal notation:

```
crash> net -N 1041236234
10.1.16.62
```

From "foreach", find all tasks with references to socket c08ea3cc:

```
crash> foreach net -s -R c08ea3cc
PID: 2184   TASK: c7026000  CPU: 1   COMMAND: "klines.kss"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 2200   TASK: c670a000  CPU: 1   COMMAND: "kpanel"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 2201   TASK: c648a000  CPU: 1   COMMAND: "kbgndwm"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 19294  TASK: c250a000  CPU: 0   COMMAND: "prefdm"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 2194   TASK: c62dc000  CPU: 1   COMMAND: "kaudioserver"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 2195   TASK: c6684000  CPU: 1   COMMAND: "maudio"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 2196   TASK: c6b58000  CPU: 1   COMMAND: "kwmsound"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 2197   TASK: c6696000  CPU: 0   COMMAND: "kfm"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 2199   TASK: c65ec000  CPU: 0   COMMAND: "krootwm"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 694    TASK: c1942000  CPU: 0   COMMAND: "prefdm"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 698    TASK: c6a2c000  CPU: 1   COMMAND: "X"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0

PID: 2159   TASK: c4a5a000  CPU: 1   COMMAND: "kwm"
FD   SOCKET     SOCK    FAMILY:TYPE         SOURCE-PORT     DESTINATION-PORT
 5  c08ea3cc  c50d3c80  INET:STREAM        0.0.0.0-1026          0.0.0.0-0
```

```
NAME
  set - set a process context or internal crash variable

SYNOPSIS
  set [[-a] [pid | taskp] | [-c cpu] | -p] | [crash_variable [setting]] | -v

DESCRIPTION
  This command either sets a new context, or gets the current context for
  display.  The context can be set by the use of:

      pid  a process PID.
    taskp  a hexadecimal task_struct pointer.
       -a  sets the pid or task as the active task on its cpu (dumpfiles only).
    -c cpu  sets the context to the active task on a cpu (dumpfiles only).
       -p  sets the context to the panic task, or back to the crash task on
           a live system.
       -v  display the current state of internal crash variables.

  If no argument is entered, the current context is displayed.  The context
  consists of the PID, the task pointer, the CPU, and task state.  The task
  state shows the bits found in both the task_struct state and exit_state
  fields.

  This command may also be used to set internal crash variables.  If no value
  argument is entered, the current value of the crash variable is shown.  These
  are the crash variables, acceptable arguments, and purpose:

         scroll  on | off     controls output scrolling.
         scroll  less         /usr/bin/less as the output scrolling program.
         scroll  more         /bin/more as the output scrolling program.
         scroll  CRASHPAGER   use CRASHPAGER environment variable as the
                              output scrolling program.
          radix  10 | 16      sets output radix to 10 or 16.
        refresh  on | off     controls internal task list refresh.
      print_max  number       set maximum number of array elements to print.
    print_array  on | off     if on, set gdb's printing of arrays to "pretty"
                              format, with one line per element.
        console  device-name  sets debug console device.
          debug  number       sets crash debug level.
           core  on | off     if on, drops core when the next error message
                              is displayed.
           hash  on | off     controls internal list verification.
         silent  on | off     turns off initialization messages; turns off
                              crash prompt during input file execution.
                              (scrolling is turned off if silent is on)
           edit  vi | emacs   set line editing mode (from .crashrc file only).
       namelist  filename     name of kernel (from .crashrc file only).
   zero_excluded  on | off    controls whether excluded pages, or pages that
                              are missing from an incomplete dumpfile, should
                              return zero-filled memory when read.
      null-stop  on | off     if on, gdb's printing of character arrays will
                              stop at the first NULL encountered.
            gdb  on | off     if on, the crash session will be run in a mode
                              where all commands will be passed directly to
                              gdb, and the command prompt will change to
                              "gdb>"; when running in this mode, native crash
                              commands may be executed by preceding them with
                              the "crash" directive.
          scope  text-addr    sets the text scope for viewing the definition
                              of data structures; the "text-addr" argument
                              must be a kernel or module text address, which
                              may be expressed symbolically or as a hexadecimal
                              value; set scope 0 to un-set.
        offline  show | hide  show or hide command output that is associated
                              with offline cpus.
        redzone  on | off     if on, CONFIG_SLUB object addresses displayed by
                              the kmem command will point to the SLAB_RED_ZONE
                              padding inserted at the beginning of the object.
   error  default | redirect | filename   set the destination of error messages.
                              "default": error messages are always displayed
                                on the console; if the output of a command is
```

```
                                    piped to an external command or redirected
                                    to a file, the error messages are also sent
                                    to the pipe or file.
                                  "redirect": if the output of a command is piped
                                    to an external command or redirected to a file,
                                    error messages are only sent to the pipe or
                                    file; otherwise they are displayed on the
                                    console.
                                  "filename": error messages are only sent to the
                                    specified filename; they are not displayed on
                                    the console and are not sent to a pipe or file.

    Internal variables may be set in four manners:

      1. entering the set command in $HOME/.crashrc.
      2. entering the set command in .crashrc in the current directory.
      3. executing an input file containing the set command.
      4. during runtime with this command.

    During initialization, $HOME/.crashrc is read first, followed by the
    .crashrc file in the current directory.  Set commands in the .crashrc file
    in the current directory override those in $HOME/.crashrc.  Set commands
    entered with this command or by runtime input file override those
    defined in either .crashrc file.  Multiple set command arguments or argument
    pairs may be entered in one command line.

EXAMPLES
    Set the current context to task c2fe8000:

      crash> set c2fe8000
           PID: 15917
       COMMAND: "bash"
          TASK: c2fe8000
           CPU: 0
         STATE: TASK_INTERRUPTIBLE

    Set the context back to the panicking task:

      crash> set -p
           PID: 698
       COMMAND: "gen12"
          TASK: f9d78000
           CPU: 2
         STATE: TASK_RUNNING (PANIC)

    Turn off output scrolling:

      crash> set scroll off
      scroll: off (/usr/bin/less)

    Show the current state of crash internal variables:

      crash> set -v
              scroll: on (/usr/bin/less)
               radix: 10 (decimal)
             refresh: on
           print_max: 256
         print_array: off
             console: /dev/pts/2
               debug: 0
                core: off
                hash: on
              silent: off
                edit: vi
           namelist: vmlinux
      zero_excluded: off
          null-stop: on
                gdb: off
              scope: (not set)
            offline: show
            redzone: on
              error: default
```

Show the current context:

```
crash> set
     PID: 1525
 COMMAND: "bash"
    TASK: c1ede000
     CPU: 0
   STATE: TASK_INTERRUPTIBLE
```

NAME

vm - virtual memory

SYNOPSIS

```
vm [-p | -P vma | -M mm | -v | -m | -x | -d | [-R reference] [pid | task]]
   [-f vm_flags]
```

DESCRIPTION

This command displays basic virtual memory information of a context, consisting of a pointer to its mm_struct and page directory, its RSS and total virtual memory size; and a list of pointers to each vm_area_struct, its starting and ending address, vm_flags value, and file pathname. If no arguments are entered, the current context is used. Additionally, the -p option translates each virtual page of each VM area to its physical address. The -R option, typically invoked from "foreach vm", searches for references to a supplied number, address, or filename argument, and prints only the essential information leading up to and including the reference. Alternatively, the -m or -v options may be used to dump the task's mm_struct or all of its vm_area_structs respectively. The -p, -v, -m, -R and -f options are all mutually exclusive.

```
          -p  translate each virtual page to its physical address, or if
              the page is not mapped, its swap device and offset, or
              filename and offset.
      -P vma  similar to -p, but only translate the pages belonging to the
              specified VM area of a context.
       -M mm  if the mm_struct address has been removed from the task_struct
              of an exiting task, the virtual memory data cannot be displayed.
              However, if the address can be determined from the kernel stack,
              it can be entered manually in order to try to resurrect the
              virtual memory data of the task.
-R reference  search for references to this number or filename.
          -m  dump the mm_struct associated with the task.
          -v  dump all of the vm_area_structs associated with the task.
          -x  override the default output format for the -m or -v options
              with hexadecimal format.
          -d  override the default output format for the -m or -v options
              with decimal format.
 -f vm_flags  translate the bits of a FLAGS (vm_flags) value.
         pid  a process PID.
        task  a hexadecimal task_struct pointer.
```

EXAMPLES

Display the virtual memory data of the current context:

```
crash> vm
PID: 30986  TASK: c0440000  CPU: 0   COMMAND: "bash"
   MM       PGD       RSS    TOTAL_VM
c303fe20  c4789000    88k      1728k
  VMA      START      END     FLAGS  FILE
c0d1f540   8048000   80ad000  1875   /bin/bash
c0d1f400   80ad000   80b3000  1873   /bin/bash
c0d1f880   80b3000   80ec000    77
c0d1f0c0  40000000  40012000   875   /lib/ld-2.1.1.so
c0d1f700  40012000  40013000   873   /lib/ld-2.1.1.so
c0d1fe00  40013000  40014000    77
c0d1f580  40014000  40016000    73
c0d1f280  4001a000  4004b000    75   /usr/lib/libncurses.so.4.2
c0d1f100  4004b000  40054000    73   /usr/lib/libncurses.so.4.2
c0d1f600  40054000  40057000    73
c0d1f9c0  40057000  40059000    75   /lib/libdl-2.1.1.so
c0d1f800  40059000  4005a000    73   /lib/libdl-2.1.1.so
```

```
  c0d1fd00  4005a000  40140000    75   /lib/libc-2.1.1.so
  c0d1fe40  40140000  40145000    73   /lib/libc-2.1.1.so
  c0d1f780  40145000  40148000    73
  c0d1f140  40148000  40150000    75   /lib/libnss_files-2.1.1.so
  c0d1fa80  40150000  40151000    73   /lib/libnss_files-2.1.1.so
  c0d1fb00  40151000  4015a000    75   /lib/libnss_nisplus-2.1.1.so
  c5f754e0  4015a000  4015b000    73   /lib/libnss_nisplus-2.1.1.so
  c0d1fec0  4015b000  4016d000    75   /lib/libnsl-2.1.1.so
  c5f75460  4016d000  4016e000    73   /lib/libnsl-2.1.1.so
  c5f75420  4016e000  40170000    73
  c5f753e0  40170000  40178000    75   /lib/libnss_nis-2.1.1.so
  c5f753a0  40178000  40179000    73   /lib/libnss_nis-2.1.1.so
  c0d1f240  bfffc000  c0000000   177
```

Display the virtual memory data along with page translations for PID 386:

```
  crash> vm -p 386
  PID: 386    TASK: c11cc000  CPU: 0   COMMAND: "atd"
     MM       PGD       RSS    TOTAL_VM
  c7e30560  c10e5000    104k     1112k
    VMA      START      END     FLAGS  FILE
  c0fbe6a0   8048000   804b000  1875   /usr/sbin/atd
   VIRTUAL  PHYSICAL
   8048000  20e1000
   8049000  17c6000
   804a000  1f6f000
    VMA      START      END     FLAGS  FILE
  c61e0ba0   804b000   804d000  1873   /usr/sbin/atd
   VIRTUAL  PHYSICAL
   804b000  254d000
   804c000  6a9c000
    VMA      START      END     FLAGS  FILE
  c61e04e0   804d000   8050000    77
   VIRTUAL  PHYSICAL
   804d000  219d000
   804e000  2617000
   804f000  SWAP: /dev/sda8  OFFSET: 24225
    VMA      START      END     FLAGS  FILE
  c61e0720  40000000  40012000   875   /lib/ld-2.1.1.so
   VIRTUAL  PHYSICAL
  40000000  FILE: /lib/ld-2.1.1.so  OFFSET: 0
  40001000  FILE: /lib/ld-2.1.1.so  OFFSET: 1000
  40002000  FILE: /lib/ld-2.1.1.so  OFFSET: 2000
  40003000  FILE: /lib/ld-2.1.1.so  OFFSET: 3000
  40004000  FILE: /lib/ld-2.1.1.so  OFFSET: 4000
  40005000  FILE: /lib/ld-2.1.1.so  OFFSET: 5000
  ...
```

Although the -R option is typically invoked from "foreach vm", it can be executed directly. This example displays all VM areas with vm_flags of 75:

```
  crash> vm -R 75
  PID: 694    TASK: c0c76000  CPU: 1   COMMAND: "crash"
     MM       PGD      RSS    TOTAL_VM
  c6c43110  c0fe9000  8932k    10720k
    VMA       START      END   FLAGS  FILE
  c322c0d0  40019000  4004a000    75  /usr/lib/libncurses.so.4.2
  c67537c0  40056000  40071000    75  /lib/libm-2.1.1.so
  c6753d00  40072000  40074000    75  /lib/libdl-2.1.1.so
  c6753540  40075000  40081000    75  /usr/lib/libz.so.1.1.3
  c6753740  40085000  4016b000    75  /lib/libc-2.1.1.so
```

One reason to use -R directly is to pare down the output associated with the -p option on a task with a huge address space. This example displays the page data associated with virtual address 40121000:

```
  crash> vm -R 40121000
  PID: 694    TASK: c0c76000  CPU: 0   COMMAND: "crash"
     MM       PGD      RSS    TOTAL_VM
  c6c43110  c0fe9000  8928k    10720k
    VMA       START      END   FLAGS  FILE
  c6753740  40085000  4016b000    75  /lib/libc-2.1.1.so
  VIRTUAL   PHYSICAL
```

```
  40121000  FILE: /lib/libc-2.1.1.so  OFFSET: 9c000

Display the mm_struct for PID 4777:

  crash> vm -m 4777
  PID: 4777   TASK: c0896000  CPU: 0   COMMAND: "bash"
  struct mm_struct {
    mmap = 0xc6caa1c0,
    mmap_avl = 0x0,
    mmap_cache = 0xc6caabc0,
    pgd = 0xc100a000,
    count = {
      counter = 0x1
    },
    map_count = 0x14,
    mmap_sem = {
      count = {
        counter = 0x1
      },
      waking = 0x0,
      wait = 0x0
    },
    context = 0x0,
    start_code = 0x8048000,
    end_code = 0x809c6f7,
    start_data = 0x0,
    end_data = 0x80a2090,
    start_brk = 0x80a5420,
    brk = 0x80b9000,
    start_stack = 0xbffff9d0,
    arg_start = 0xbffffad1,
    arg_end = 0xbffffad7,
    env_start = 0xbffffad7,
    env_end = 0xbfffffe2,
    rss = 0xf6,
    total_vm = 0x1a3,
    locked_vm = 0x0,
    def_flags = 0x0,
    cpu_vm_mask = 0x0,
    swap_cnt = 0x23d,
    swap_address = 0x0,
    segments = 0x0
  }

Display all of the vm_area_structs for task c47d4000:

  crash> vm -v c47d4000
  PID: 4971   TASK: c47d4000  CPU: 1   COMMAND: "login"
  struct vm_area_struct {
    vm_mm = 0xc4b0d200,
    vm_start = 0x8048000,
    vm_end = 0x804d000,
    vm_next = 0xc3e3abd0,
    vm_page_prot = {
      pgprot = 0x25
    },
    vm_flags = 0x1875,
    vm_avl_height = 0x1,
    vm_avl_left = 0x0,
    vm_avl_right = 0x0,
    vm_next_share = 0x0,
    vm_pprev_share = 0xc3e3abf0,
    vm_ops = 0xc02392a0,
    vm_offset = 0x0,
    vm_file = 0xc1e23660,
    vm_pte = 0x0
  }
  struct vm_area_struct {
    vm_mm = 0xc4b0d200,
    vm_start = 0x804d000,
    vm_end = 0x804e000,
    vm_next = 0xc3e3a010,
    vm_page_prot = {
```

```
          pgprot = 0x25
        },
        vm_flags = 0x1873,
        vm_avl_height = 0x2,
        vm_avl_left = 0xc3e3a810,
        vm_avl_right = 0xc3e3a010,
        vm_next_share = 0xc3e3a810,
        vm_pprev_share = 0xc3699c14
        ...

  Translate a FLAGS value:

    crash> vm -f 3875
    3875: (READ|EXEC|MAYREAD|MAYWRITE|MAYEXEC|DENYWRITE|EXECUTABLE|LOCKED)

  Display the page translations of the VM area at address f5604f2c:

    crash> vm -P f5604f2c
    PID: 5508   TASK: f56a9570  CPU: 0   COMMAND: "crond"
      VMA       START      END    FLAGS  FILE
    f5604f2c    f5b000    f67000 8000075  /lib/libnss_files-2.12.so
    VIRTUAL   PHYSICAL
    f5b000    3fec1000
    f5c000    3d3a4000
    f5d000    FILE: /lib/libnss_files-2.12.so  OFFSET: 2000
    f5e000    FILE: /lib/libnss_files-2.12.so  OFFSET: 3000
    f5f000    FILE: /lib/libnss_files-2.12.so  OFFSET: 4000
    f60000    3fd31000
    f61000    3fd32000
    f62000    FILE: /lib/libnss_files-2.12.so  OFFSET: 7000
    f63000    FILE: /lib/libnss_files-2.12.so  OFFSET: 8000
    f64000    3ff35000
    f65000    FILE: /lib/libnss_files-2.12.so  OFFSET: a000
    f66000    FILE: /lib/libnss_files-2.12.so  OFFSET: b000


NAME
  bpf - extended Berkeley Packet Filter (eBPF)

SYNOPSIS
  bpf [[-p ID | -P] [-tTj]] [[-m ID] | -M] [-s] [-xd]

DESCRIPTION

  This command provides information on currently-loaded eBPF programs and maps.
  With no arguments, basic information about each loaded eBPF program and map
  is displayed.  For each eBPF program, its ID number, the addresses of its
  bpf_prog and bpf_prog_aux data structures, its type, tag, and the IDs of the
  eBPF maps that it uses are displayed.  For each eBPF map, its ID number, the
  address of its bpf_map data structure, its type, and the hexadecimal value of
  its map_flags are displayed.

    -p ID  displays the basic information specific to the program ID, plus the
           size in bytes of its translated bytecode, the size in bytes of its
           jited code, the number of bytes locked into memory, the time that
           the program was loaded, whether it is GPL compatible, its name
           string, and its UID.
    -P     same as -p, but displays the basic and extra data for all programs.
    -m ID  displays the basic information specific to the map ID, plus the
           size in bytes of its key and value, the maximum number of key-value
           pairs that can be stored within the map, the number of bytes locked
           into memory, its name string, and its UID.
    -M     same as -m, but displays the basic and extra data for all maps.
    -t     translate the bytecode of the specified program ID.
    -T     same as -t, but also dump the bytecode of each instruction.
    -j     disassemble the jited code of the specified program ID.
    -s     with -p or -P, dump the bpf_prog and bpf_prog_aux data structures.
           with -m or -M, dump the bpf_map structure.
    -x     with -s, override default output format with hexadecimal format.
    -d     with -s, override default output format with decimal format.

EXAMPLES
  Display all loaded eBPF programs and maps:
```

```
crash> bpf
 ID     BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG         USED_MAPS
 13 ffffbc00c06d1000 ffff9ff260f0c400  CGROUP_SKB   7be49e3934a125ba   13,14
 14 ffffbc00c0761000 ffff9ff260f0f600  CGROUP_SKB   2a142ef67aaad174   13,14
 15 ffffbc00c001d000 ffff9ff2618f9e00  CGROUP_SKB   7be49e3934a125ba   15,16
 16 ffffbc00c06c9000 ffff9ff2618f9400  CGROUP_SKB   2a142ef67aaad174   15,16
 19 ffffbc00c0d39000 ffff9ff2610fa000  CGROUP_SKB   7be49e3934a125ba   19,20
 20 ffffbc00c0d41000 ffff9ff2610f8e00  CGROUP_SKB   2a142ef67aaad174   19,20
 30 ffffbc00c065f000 ffff9ff1b64de200    KPROBE     69fed6de18629d7a    32
 31 ffffbc00c065b000 ffff9ff1b64df200    KPROBE     69fed6de18629d7a    37
 32 ffffbc00c0733000 ffff9ff1b64dc600    KPROBE     69fed6de18629d7a    38
 33 ffffbc00c0735000 ffff9ff1b64dca00    KPROBE     69fed6de18629d7a    39
 34 ffffbc00c0737000 ffff9ff1b64dfc00    KPROBE     4abbddae72a6ee17 33,36,34
 36 ffffbc00c0839000 ffff9ff1b64dd000    KPROBE     da4fc6a3f41761a2    32
 41 ffffbc00c07ec000 ffff9ff207b70400  TRACEPOINT   e2094f9f46284bf6   55,54
 44 ffffbc00c07ee000 ffff9ff1b64dc800  PERF_EVENT   19578a12836c4115    62
 46 ffffbc00c07f0000 ffff9ff207b70400 SOCKET_FILTER 1fcfc04afd689133    64

 ID     BPF_MAP       BPF_MAP_TYPE   MAP_FLAGS
 13 ffff9ff260f0ec00    LPM_TRIE      00000001
 14 ffff9ff260f0de00    LPM_TRIE      00000001
 15 ffff9ff2618fbe00    LPM_TRIE      00000001
 16 ffff9ff2618fb800    LPM_TRIE      00000001
 19 ffff9ff2610faa00    LPM_TRIE      00000001
 20 ffff9ff2610fb800    LPM_TRIE      00000001
 32 ffff9ff260d74000      HASH        00000000
 33 ffff9ff260d76400    LRU_HASH      00000000
 34 ffff9ff260d70000    LRU_HASH      00000002
 35 ffff9ff260d73800    LRU_HASH      00000004
 36 ffff9ff1b4f44000  ARRAY_OF_MAPS   00000000
 37 ffff9ff260d77c00   PERCPU_HASH    00000000
 38 ffff9ff260d70800      HASH        00000001
 39 ffff9ff260d76c00   PERCPU_HASH    00000001
 54 ffff9ff260dd2c00      HASH        00000000
 55 ffff9ff260dd1400      HASH        00000000
 62 ffff9ff1ae784000      HASH        00000000
 64 ffff9ff1aea15000      ARRAY       00000000

Display additional data about program ID 20:

crash> bpf -p 20
 ID     BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG         USED_MAPS
 20 ffffbc00c0d41000 ffff9ff2610f8e00  CGROUP_SKB   2a142ef67aaad174   19,20
    XLATED: 296  JITED: 229  MEMLOCK: 4096
    LOAD_TIME: Fri Apr 20 19:39:21 2018
    GPL_COMPATIBLE: yes  UID: 0

Display additional data about map ID 34:

crash> bpf -m 34
 ID     BPF_MAP       BPF_MAP_TYPE   MAP_FLAGS
 34  ffff9ff260d70000    LRU_HASH      00000000
     KEY_SIZE: 4  VALUE_SIZE: 8  MAX_ENTRIES: 10000  MEMLOCK: 1953792
     NAME: "lru_hash_map"  UID: 0

Disassemble the jited program of program ID 20:

crash> bpf -p 20 -j
ID     BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG         USED_MAPS
20 ffffbc00c0d41000 ffff9ff2610f8e00  CGROUP_SKB   2a142ef67aaad174   19,20
   XLATED: 296  JITED: 229  MEMLOCK: 4096
   LOAD_TIME: Fri Apr 20 19:39:21 2018
   GPL_COMPATIBLE: yes  UID: 0

 0xffffffffc06887a2:  push   %rbp
 0xffffffffc06887a3:  mov    %rsp,%rbp
 0xffffffffc06887a6:  sub    $0x40,%rsp
 0xffffffffc06887ad:  sub    $0x28,%rbp
 0xffffffffc06887b1:  mov    %rbx,0x0(%rbp)
 0xffffffffc06887b5:  mov    %r13,0x8(%rbp)
 0xffffffffc06887b9:  mov    %r14,0x10(%rbp)
 0xffffffffc06887bd:  mov    %r15,0x18(%rbp)
```

```
   0xffffffffc06887c1:  xor    %eax,%eax
   0xffffffffc06887c3:  mov    %rax,0x20(%rbp)
   0xffffffffc06887c7:  mov    %rdi,%rbx
   0xffffffffc06887ca:  movzwq 0xc0(%rbx),%r13
   0xffffffffc06887d2:  xor    %r14d,%r14d
   0xffffffffc06887d5:  cmp    $0x8,%r13
   0xffffffffc06887d9:  jne    0xffffffffc068881b
   0xffffffffc06887db:  mov    %rbx,%rdi
   0xffffffffc06887de:  mov    $0xc,%esi
   0xffffffffc06887e3:  mov    %rbp,%rdx
   0xffffffffc06887e6:  add    $0xfffffffffffffffc,%rdx
   0xffffffffc06887ea:  mov    $0x4,%ecx
   0xffffffffc06887ef:  callq  0xffffffffb0865340 <bpf_skb_load_bytes>
   0xffffffffc06887f4:  movabs $0xffff9ff2610faa00,%rdi
   0xffffffffc06887fe:  mov    %rbp,%rsi
   0xffffffffc0688801:  add    $0xfffffffffffffff8,%rsi
   0xffffffffc0688805:  movl   $0x20,0x0(%rsi)
   0xffffffffc068880c:  callq  0xffffffffb01fcba0 <bpf_map_lookup_elem>
   0xffffffffc0688811:  cmp    $0x0,%rax
   0xffffffffc0688815:  je     0xffffffffc068881b
   0xffffffffc0688817:  or     $0x2,%r14d
   0xffffffffc068881b:  cmp    $0xdd86,%r13
   0xffffffffc0688822:  jne    0xffffffffc0688864
   0xffffffffc0688824:  mov    %rbx,%rdi
   0xffffffffc0688827:  mov    $0x8,%esi
   0xffffffffc068882c:  mov    %rbp,%rdx
   0xffffffffc068882f:  add    $0xfffffffffffffff0,%rdx
   0xffffffffc0688833:  mov    $0x10,%ecx
   0xffffffffc0688838:  callq  0xffffffffb0865340 <bpf_skb_load_bytes>
   0xffffffffc068883d:  movabs $0xffff9ff2610fb800,%rdi
   0xffffffffc0688847:  mov    %rbp,%rsi
   0xffffffffc068884a:  add    $0xffffffffffffffec,%rsi
   0xffffffffc068884e:  movl   $0x80,0x0(%rsi)
   0xffffffffc0688855:  callq  0xffffffffb01fcba0 <bpf_map_lookup_elem>
   0xffffffffc068885a:  cmp    $0x0,%rax
   0xffffffffc068885e:  je     0xffffffffc0688864
   0xffffffffc0688860:  or     $0x2,%r14d
   0xffffffffc0688864:  mov    $0x1,%eax
   0xffffffffc0688869:  cmp    $0x2,%r14
   0xffffffffc068886d:  jne    0xffffffffc0688871
   0xffffffffc068886f:  xor    %eax,%eax
   0xffffffffc0688871:  mov    0x0(%rbp),%rbx
   0xffffffffc0688875:  mov    0x8(%rbp),%r13
   0xffffffffc0688879:  mov    0x10(%rbp),%r14
   0xffffffffc068887d:  mov    0x18(%rbp),%r15
   0xffffffffc0688881:  add    $0x28,%rbp
   0xffffffffc0688885:  leaveq
   0xffffffffc0688886:  retq

Translate each bytecode instruction of program ID 13:

crash> bpf -p 13 -t
 ID      BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG        USED_MAPS
 13 ffffbc00c06d1000 ffff9ff260f0c400  CGROUP_SKB   7be49e3934a125ba   13,14
    XLATED: 296  JITED: 229  MEMLOCK: 4096
    LOAD_TIME: Fri Apr 20 19:39:11 2018
    GPL_COMPATIBLE: yes  UID: 0

  0: (bf) r6 = r1
  1: (69) r7 = *(u16 *)(r6 +192)
  2: (b4) (u32) r8 = (u32) 0
  3: (55) if r7 != 0x8 goto pc+14
  4: (bf) r1 = r6
  5: (b4) (u32) r2 = (u32) 16
  6: (bf) r3 = r10
  7: (07) r3 += -4
  8: (b4) (u32) r4 = (u32) 4
  9: (85) call bpf_skb_load_bytes#6793152
 10: (18) r1 = map[id:13]
 12: (bf) r2 = r10
 13: (07) r2 += -8
 14: (62) *(u32 *)(r2 +0) = 32
 15: (85) call bpf_map_lookup_elem#73760
```

```
 16: (15) if r0 == 0x0 goto pc+1
 17: (44) (u32) r8 |= (u32) 2
 18: (55) if r7 != 0xdd86 goto pc+14
 19: (bf) r1 = r6
 20: (b4) (u32) r2 = (u32) 24
 21: (bf) r3 = r10
 22: (07) r3 += -16
 23: (b4) (u32) r4 = (u32) 16
 24: (85) call bpf_skb_load_bytes#6793152
 25: (18) r1 = map[id:14]
 27: (bf) r2 = r10
 28: (07) r2 += -20
 29: (62) *(u32 *)(r2 +0) = 128
 30: (85) call bpf_map_lookup_elem#73760
 31: (15) if r0 == 0x0 goto pc+1
 32: (44) (u32) r8 |= (u32) 2
 33: (b7) r0 = 1
 34: (55) if r8 != 0x2 goto pc+1
 35: (b7) r0 = 0
 36: (95) exit
```

Translate, and then dump each bytecode instruction of program ID 13:

```
crash> bpf -p 13 -T
 ID      BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG       USED_MAPS
 13 ffffbc00c06d1000 ffff9ff260f0c400  CGROUP_SKB   7be49e3934a125ba   13,14
    XLATED: 296  JITED: 229  MEMLOCK: 4096
    LOAD_TIME: Fri Apr 20 19:39:11 2018
    GPL_COMPATIBLE: yes  UID: 0

  0: (bf) r6 = r1
      bf 16 00 00 00 00 00 00
  1: (69) r7 = *(u16 *)(r6 +192)
      69 67 c0 00 00 00 00 00
  2: (b4) (u32) r8 = (u32) 0
      b4 08 00 00 00 00 00 00
  3: (55) if r7 != 0x8 goto pc+14
      55 07 0e 00 08 00 00 00
  4: (bf) r1 = r6
      bf 61 00 00 00 00 00 00
  5: (b4) (u32) r2 = (u32) 16
      b4 02 00 00 10 00 00 00
  6: (bf) r3 = r10
      bf a3 00 00 00 00 00 00
  7: (07) r3 += -4
      07 03 00 00 fc ff ff ff
  8: (b4) (u32) r4 = (u32) 4
      b4 04 00 00 04 00 00 00
  9: (85) call bpf_skb_load_bytes#6793152
      85 00 00 00 c0 a7 67 00
 10: (18) r1 = map[id:13]
      18 01 00 00 00 7a 96 61 00 00 00 00 b2 9d ff ff
 12: (bf) r2 = r10
      bf a2 00 00 00 00 00 00
 13: (07) r2 += -8
      07 02 00 00 f8 ff ff ff
 14: (62) *(u32 *)(r2 +0) = 32
      62 02 00 00 20 00 00 00
 15: (85) call bpf_map_lookup_elem#73760
      85 00 00 00 20 20 01 00
 16: (15) if r0 == 0x0 goto pc+1
      15 00 01 00 00 00 00 00
 17: (44) (u32) r8 |= (u32) 2
      44 08 00 00 02 00 00 00
 18: (55) if r7 != 0xdd86 goto pc+14
      55 07 0e 00 86 dd 00 00
 19: (bf) r1 = r6
      bf 61 00 00 00 00 00 00
 20: (b4) (u32) r2 = (u32) 24
      b4 02 00 00 18 00 00 00
 21: (bf) r3 = r10
      bf a3 00 00 00 00 00 00
 22: (07) r3 += -16
```

```
           07 03 00 00 f0 ff ff ff
     23: (b4) (u32) r4 = (u32) 16
           b4 04 00 00 10 00 00 00
     24: (85) call bpf_skb_load_bytes#6793152
           85 00 00 00 c0 a7 67 00
     25: (18) r1 = map[id:14]
           18 01 00 00 00 68 96 61 00 00 00 00 b2 9d ff ff
     27: (bf) r2 = r10
           bf a2 00 00 00 00 00 00
     28: (07) r2 += -20
           07 02 00 00 ec ff ff ff
     29: (62) *(u32 *)(r2 +0) = 128
           62 02 00 00 80 00 00 00
     30: (85) call bpf_map_lookup_elem#73760
           85 00 00 00 20 20 01 00
     31: (15) if r0 == 0x0 goto pc+1
           15 00 01 00 00 00 00 00
     32: (44) (u32) r8 |= (u32) 2
           44 08 00 00 02 00 00 00
     33: (b7) r0 = 1
           b7 00 00 00 01 00 00 00
     34: (55) if r8 != 0x2 goto pc+1
           55 08 01 00 02 00 00 00
     35: (b7) r0 = 0
           b7 00 00 00 00 00 00 00
     36: (95) exit
           95 00 00 00 00 00 00 00
```

Display the bpf_map data structure for map ID 13:

```
crash> bpf -m 13 -s
 ID      BPF_MAP       BPF_MAP_TYPE   MAP_FLAGS
 13  ffff9ff260f0ec00    LPM_TRIE      00000001
     KEY_SIZE: 8  VALUE_SIZE: 8  MAX_ENTRIES: 1  MEMLOCK: 4096
     NAME: (unused)  UID: 0

struct bpf_map {
  ops = 0xffffffffb0e36720,
  inner_map_meta = 0x0,
  security = 0xffff9ff26873a158,
  map_type = BPF_MAP_TYPE_LPM_TRIE,
  key_size = 8,
  value_size = 8,
  max_entries = 1,
  map_flags = 1,
  pages = 1,
  id = 13,
  numa_node = -1,
  unpriv_array = false,
  user = 0xffffffffb14578a0,
  refcnt = {
    counter = 3
  },
  usercnt = {
    counter = 1
  },
  work = {
    data = {
      counter = 0
    },
    entry = {
      next = 0x0,
      prev = 0x0
    },
    func = 0x0,
    lockdep_map = {
      key = 0x0,
      class_cache = {0x0, 0x0},
      name = 0x0,
      cpu = 0,
      ip = 0
    }
  },
```

```
  name = "
}

Display the bpf_prog and bpf_prog_aux structures for program ID 13:

crash> bpf -p 13 -s
 ID      BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG       USED_MAPS
 13  ffffbc00c06d1000 ffff9ff260f0c400  CGROUP_SKB   7be49e3934a125ba   13,14
     XLATED: 296  JITED: 229  MEMLOCK: 4096
     LOAD_TIME: Fri Apr 20 19:39:10 2018
     GPL_COMPATIBLE: yes  UID: 0

 struct bpf_prog {
   pages = 1,
   jited = 1,
   jit_requested = 1,
   locked = 1,
   gpl_compatible = 1,
   cb_access = 0,
   dst_needed = 0,
   blinded = 0,
   is_func = 0,
   kprobe_override = 0,
   type = BPF_PROG_TYPE_CGROUP_SKB,
   len = 37,
   jited_len = 229,
   tag = "{\344\236\071\064\241%\272",
   aux = ffff9ff260f0c400,
   orig_prog = 0x0,
   bpf_func = 0xffffffffc0218a59,
   {
     insns = 0xffffb0cf406d1030,
     insnsi = 0xffffb0cf406d1030
   }
 }

 struct bpf_prog_aux {
   refcnt = {
     counter = 2
   },
   used_map_cnt = 2,
   max_ctx_offset = 20,
   stack_depth = 20,
   id = 13,
   func_cnt = 0,
   offload_requested = false,
   func = 0x0,
   jit_data = 0x0,
   ksym_tnode = {
     node = {{
          __rb_parent_color = 18446635988194065457,
          rb_right = 0x0,
          rb_left = 0x0
        }, {
          __rb_parent_color = 18446635988194065481,
          rb_right = 0x0,
          rb_left = 0x0
        }}
   },
   ksym_lnode = {
     next = 0xffff9db261966460,
     prev = 0xffffffffb85d1150
   },
   ops = 0xffffffffb7f09060,
   used_maps = 0xffff9db261e03600,
   prog = 0xffffb0cf406d1000,
   user = 0xffffffffb84578a0,
   load_time = 23962237943,
   name = "
   security = 0xffff9db266f9cf50,
   offload = 0x0,
   {
     work = {
```

```
          data = {
            counter = 0
          },
          entry = {
            next = 0x0,
            prev = 0x0
          },
          func = 0x0,
          lockdep_map = {
            key = 0x0,
            class_cache = {0x0, 0x0},
            name = 0x0,
            cpu = 0,
            ip = 0
          }
        },
        rcu = {
          next = 0x0,
          func = 0x0
        }
      }
    }
```

Display the extra data about all programs:

```
crash> bpf -P
 ID     BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG         USED_MAPS
 13 ffffbc00c06d1000 ffff9ff260f0c400  CGROUP_SKB   7be49e3934a125ba   13,14
    XLATED: 296  JITED: 229  MEMLOCK: 4096
    LOAD_TIME: Fri Apr 20 19:39:10 2018
    GPL_COMPATIBLE: yes  UID: 0

 ID     BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG         USED_MAPS
 14 ffffbc00c0761000 ffff9ff260f0f600  CGROUP_SKB   2a142ef67aaad174   13,14
    XLATED: 296  JITED: 229  MEMLOCK: 4096
    LOAD_TIME: Fri Apr 20 19:39:10 2018
    GPL_COMPATIBLE: yes  UID: 0

 ID     BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG         USED_MAPS
 15 ffffbc00c001d000 ffff9ff2618f9e00  CGROUP_SKB   7be49e3934a125ba   15,16
    XLATED: 296  JITED: 229  MEMLOCK: 4096
    LOAD_TIME: Fri Apr 20 19:39:11 2018
    GPL_COMPATIBLE: yes  UID: 0

...

 ID     BPF_PROG       BPF_PROG_AUX   BPF_PROG_TYPE       TAG         USED_MAPS
 75 ffffbc00c0ed1000 ffff9ff2429c6400    KPROBE     da4fc6a3f41761a2    107
    XLATED: 5168  JITED: 2828  MEMLOCK: 8192
    LOAD_TIME: Fri Apr 27 14:54:40 2018
    GPL_COMPATIBLE: yes  UID: 0
```

Display the extra data for all maps:

```
crash> bpf -M
 ID      BPF_MAP       BPF_MAP_TYPE   MAP_FLAGS
 13  ffff9ff260f0ec00    LPM_TRIE      00000001
     KEY_SIZE: 8  VALUE_SIZE: 8  MAX_ENTRIES: 1  MEMLOCK: 4096
     NAME: (unused)  UID: 0

 ID      BPF_MAP       BPF_MAP_TYPE   MAP_FLAGS
 14  ffff9ff260f0de00    LPM_TRIE      00000001
     KEY_SIZE: 20  VALUE_SIZE: 8  MAX_ENTRIES: 1  MEMLOCK: 4096
     NAME: (unused)  UID: 0

...

 ID      BPF_MAP       BPF_MAP_TYPE   MAP_FLAGS
108  ffff9ff1aeab9400    LRU_HASH      00000000
     KEY_SIZE: 4  VALUE_SIZE: 8  MAX_ENTRIES: 1000  MEMLOCK: 147456
     NAME: "lru_hash_lookup"  UID: 0
```

To display all possible information that this command offers about

```
  all programs and maps, enter:

  crash> bpf -PM -jTs


NAME
  gdb - gdb command

SYNOPSIS
  gdb command ...

DESCRIPTION
  This command passes its arguments directly to gdb for processing.
  This is typically not necessary, but where ambiguities between crash and
  gdb command names exist, this will force the command to be executed by gdb.

  Alternatively, if "set gdb on" is entered, the session will be run in a
  mode where all commands are passed directly to gdb.  When running in that
  mode, native crash commands may be executed by preceding them with the
  "crash" directive.  To restore native crash mode, enter "set gdb off".

EXAMPLES
    crash> gdb help
    List of classes of commands:

    aliases -- Aliases of other commands
    breakpoints -- Making program stop at certain points
    data -- Examining data
    files -- Specifying and examining files
    internals -- Maintenance commands
    obscure -- Obscure features
    running -- Running the program
    stack -- Examining the stack
    status -- Status inquiries
    support -- Support facilities
    tracepoints -- Tracing of program execution without stopping the program
    user-defined -- User-defined commands

    Type "help" followed by a class name for a list of commands in that class.
    Type "help" followed by command name for full documentation.
    Command name abbreviations are allowed if unambiguous.


NAME
  p - print the value of an expression

SYNOPSIS
  p [-x|-d][-u] [expression | symbol[:cpuspec]]

DESCRIPTION
  This command passes its arguments on to gdb "print" command for evaluation.

    expression  an expression to be evaluated.
        symbol  a kernel symbol.
      :cpuspec  CPU specification for a per-cpu symbol:
                  :             CPU of the currently selected task.
                  :a[ll]        all CPUs.
                  :#[-#][,...]  CPU list(s), e.g. "1,3,5", "1-3",
                                or "1,3,5-7,10".
            -x  override default output format with hexadecimal format.
            -d  override default output format with decimal format.
            -u  the expression evaluates to a user address reference.

  The default output format is decimal, but that can be changed at any time
  with the two built-in aliases "hex" and "dec".  Alternatively, there
  are two other built-in aliases, "px" and "pd", which force the command
  output to be displayed in hexadecimal or decimal, without changing the
  default mode.

EXAMPLES
  Print the contents of jiffies:

    crash> p jiffies
```

```
  jiffies = $6 = 166532620
  crash> px jiffies
  jiffies = $7 = 0x9ed174b
  crash> pd jiffies
  jiffies = $8 = 166533160
```

Print the contents of the vm_area_struct "init_mm":

```
  crash> p init_mm
  init_mm = $5 = {
    mmap = 0xc022d540,
    mmap_avl = 0x0,
    mmap_cache = 0x0,
    pgd = 0xc0101000,
    count = {
      counter = 0x6
    },
    map_count = 0x1,
    mmap_sem = {
      count = {
        counter = 0x1
      },
      waking = 0x0,
      wait = 0x0
    },
    context = 0x0,
    start_code = 0xc0000000,
    end_code = 0xc022b4c8,
    start_data = 0x0,
    end_data = 0xc0250388,
    start_brk = 0x0,
    brk = 0xc02928d8,
    start_stack = 0x0,
    arg_start = 0x0,
    arg_end = 0x0,
    env_start = 0x0,
    env_end = 0x0,
    rss = 0x0,
    total_vm = 0x0,
    locked_vm = 0x0,
    def_flags = 0x0,
    cpu_vm_mask = 0x0,
    swap_cnt = 0x0,
    swap_address = 0x0,
    segments = 0x0
  }
```

If a per-cpu symbol is entered as a argument, its data type and all of its per-cpu addresses are displayed:

```
  crash> p irq_stat
  PER-CPU DATA TYPE:
    irq_cpustat_t irq_stat;
  PER-CPU ADDRESSES:
    [0]: ffff88021e211540
    [1]: ffff88021e251540
    [2]: ffff88021e291540
    [3]: ffff88021e2d1540
```

To display the contents a per-cpu symbol for CPU 1, append a cpu-specifier:

```
  crash> p irq_stat:1
  per_cpu(irq_stat, 1) = $29 = {
    __softirq_pending = 0,
    __nmi_count = 209034,
    apic_timer_irqs = 597509876,
    irq_spurious_count = 0,
    icr_read_retry_count = 2,
    x86_platform_ipis = 0,
    apic_perf_irqs = 209034,
    apic_irq_work_irqs = 0,
    irq_resched_count = 264922233,
```

```
      irq_call_count = 7036692,
      irq_tlb_count = 4750442,
      irq_thermal_count = 0,
      irq_threshold_count = 0
    }



NAME
  sig - task signal handling

SYNOPSIS
  sig [[-l] | [-s sigset]] | [-g] [pid | taskp] ...

DESCRIPTION
  This command displays signal-handling data of one or more tasks.  Multiple
  task or PID numbers may be entered; if no arguments are entered, the signal
  handling data of the current context will be displayed.  The default display
  shows:

    1.  A formatted dump of the "sig" signal_struct structure referenced by
        the task_struct.  For each defined signal, it shows the sigaction
        structure address, the signal handler, the signal sigset_t mask
        (also expressed as a 64-bit hexadecimal value), and the flags.
    2.  Whether the task has an unblocked signal pending.
    3.  The contents of the "blocked" and "signal" sigset_t structures
        from the task_struct/signal_struct, both of which are represented
        as a 64-bit hexadecimal value.
    4.  For each queued signal, private and/or shared, if any, its signal
        number and associated siginfo structure address.

  The -l option lists the signal numbers and their name(s).  The -s option
  translates a 64-bit hexadecimal value representing the contents of a
  sigset_t structure into the signal names whose bits are set.

        pid  a process PID.
      taskp  a hexadecimal task_struct pointer.
         -g  displays signal information for all threads in a task's
             thread group.
         -l  displays the defined signal numbers and names.
  -s sigset  translates a 64-bit hexadecimal value representing a sigset_t
             into a list of signal names associated with the bits set.

EXAMPLES
  Dump the signal-handling data of PID 8970:

    crash> sig 8970
    PID: 8970   TASK: f67d8560  CPU: 1   COMMAND: "procsig"
    SIGNAL_STRUCT: f6018680  COUNT: 1
     SIG SIGACTION  HANDLER       MASK       FLAGS
     [1]  f7877684  SIG_DFL 0000000000000000 0
     [2]  f7877698  SIG_DFL 0000000000000000 0
    ...
     [8]  f7877710  SIG_DFL 0000000000000000 0
     [9]  f7877724  SIG_DFL 0000000000000000 0
    [10]  f7877738  804867a 0000000000000000 80000000 (SA_RESETHAND)
    [11]  f787774c  SIG_DFL 0000000000000000 0
    [12]  f7877760  804867f 0000000000000000 10000004 (SA_SIGINFO|SA_RESTART)
    [13]  f7877774  SIG_DFL 0000000000000000 0
    ...
    [31]  f78778dc  SIG_DFL 0000000000000000 0
    [32]  f78778f0  SIG_DFL 0000000000000000 0
    [33]  f7877904  SIG_DFL 0000000000000000 0
    [34]  f7877918  804867f 0000000000000000 10000004 (SA_SIGINFO|SA_RESTART)
    [35]  f787792c  SIG_DFL 0000000000000000 0
    [36]  f7877940  SIG_DFL 0000000000000000 0
    ...
    [58]  f7877af8  SIG_DFL 0000000000000000 0
    [59]  f7877b0c  SIG_DFL 0000000000000000 0
    [60]  f7877b20  SIG_DFL 0000000000000000 0
    [61]  f7877b34  SIG_DFL 0000000000000000 0
    [62]  f7877b48  SIG_DFL 0000000000000000 0
    [63]  f7877b5c  SIG_DFL 0000000000000000 0
```

```
 [64]  f7877b70  804867f 0000000000000000 10000004 (SA_SIGINFO|SA_RESTART)
SIGPENDING: no
   BLOCKED: 8000000200000800
PRIVATE_PENDING
    SIGNAL: 0000000200000800
  SIGQUEUE:  SIG  SIGINFO
              12  f51b9c84
              34  f51b9594
SHARED_PENDING
    SIGNAL: 8000000000000800
  SIGQUEUE:  SIG  SIGINFO
              12  f51b9188
              64  f51b9d18
              64  f51b9500
```

Dump the signal-handling data for all tasks in the thread group containing PID 2578:

```
crash> sig -g 2578
PID: 2387   TASK: f617d020  CPU: 0   COMMAND: "slapd"
SIGNAL_STRUCT: f7dede00  COUNT: 6
SIG SIGACTION  HANDLER       MASK       FLAGS
[1]  c1f60c04   a258a7 0000000000000000 10000000 (SA_RESTART)
[2]  c1f60c18   a258a7 0000000000000000 10000000 (SA_RESTART)
[3]  c1f60c2c  SIG_DFL 0000000000000000 0
[4]  c1f60c40  SIG_DFL 0000000000000000 0
[5]  c1f60c54   a258a7 0000000000000000 10000000 (SA_RESTART)
[6]  c1f60c68  SIG_DFL 0000000000000000 0
[7]  c1f60c7c  SIG_DFL 0000000000000000 0
[8]  c1f60c90  SIG_DFL 0000000000000000 0
[9]  c1f60ca4  SIG_DFL 0000000000000000 0
[10]  c1f60cb8   a25911 0000000000000000 10000000 (SA_RESTART)
...
[64]  c1f610f0  SIG_DFL 0000000000000000 0
SHARED_PENDING
    SIGNAL: 0000000000000000
  SIGQUEUE: (empty)

  PID: 2387   TASK: f617d020  CPU: 0   COMMAND: "slapd"
  SIGPENDING: no
     BLOCKED: 0000000000000000
  PRIVATE_PENDING
      SIGNAL: 0000000000000000
    SIGQUEUE: (empty)

  PID: 2392   TASK: f6175aa0  CPU: 0   COMMAND: "slapd"
  SIGPENDING: no
     BLOCKED: 0000000000000000
  PRIVATE_PENDING
      SIGNAL: 0000000000000000
    SIGQUEUE: (empty)

  PID: 2523   TASK: f7cd4aa0  CPU: 1   COMMAND: "slapd"
  SIGPENDING: no
     BLOCKED: 0000000000000000
  PRIVATE_PENDING
      SIGNAL: 0000000000000000
    SIGQUEUE: (empty)

  ...
```

Translate the sigset_t mask value, cut-and-pasted from the signal handling data from signals 1 and 10 above:

```
crash> sig -s 800A000000000201
SIGHUP SIGUSR1 SIGRTMAX-14 SIGRTMAX-12 SIGRTMAX
```

List the signal numbers and their names:

```
crash> sig -l
 [1] SIGHUP
 [2] SIGINT
 [3] SIGQUIT
```

```
       [4] SIGILL
       [5] SIGTRAP
       [6] SIGABRT/SIGIOT
       [7] SIGBUS
       [8] SIGFPE
       [9] SIGKILL
      [10] SIGUSR1
      [11] SIGSEGV
      [12] SIGUSR2
      [13] SIGPIPE
      [14] SIGALRM
      [15] SIGTERM
      [16] SIGSTKFLT
      [17] SIGCHLD/SIGCLD
      [18] SIGCONT
      [19] SIGSTOP
      [20] SIGTSTP
      [21] SIGTTIN
      [22] SIGTTOU
      [23] SIGURG
      [24] SIGXCPU
      [25] SIGXFSZ
      [26] SIGVTALRM
      [27] SIGPROF
      [28] SIGWINCH
      [29] SIGIO/SIGPOLL
      [30] SIGPWR
      [31] SIGSYS
      [32] SIGRTMIN
      [33] SIGRTMIN+1
      [34] SIGRTMIN+2
      [35] SIGRTMIN+3
      [36] SIGRTMIN+4
      [37] SIGRTMIN+5
      [38] SIGRTMIN+6
      [39] SIGRTMIN+7
      [40] SIGRTMIN+8
      [41] SIGRTMIN+9
      [42] SIGRTMIN+10
      [43] SIGRTMIN+11
      [44] SIGRTMIN+12
      [45] SIGRTMIN+13
      [46] SIGRTMIN+14
      [47] SIGRTMIN+15
      [48] SIGRTMIN+16
      [49] SIGRTMAX-15
      [50] SIGRTMAX-14
      [51] SIGRTMAX-13
      [52] SIGRTMAX-12
      [53] SIGRTMAX-11
      [54] SIGRTMAX-10
      [55] SIGRTMAX-9
      [56] SIGRTMAX-8
      [57] SIGRTMAX-7
      [58] SIGRTMAX-6
      [59] SIGRTMAX-5
      [60] SIGRTMAX-4
      [61] SIGRTMAX-3
      [62] SIGRTMAX-2
      [63] SIGRTMAX-1
      [64] SIGRTMAX


NAME
  vtop - virtual to physical

SYNOPSIS
  vtop [-c [pid | taskp]] [-u|-k] address ...

DESCRIPTION
  This command translates a user or kernel virtual address to its physical
  address.  Also displayed is the PTE translation, the vm_area_struct data
  for user virtual addresses, the mem_map page data associated with the
```

```
    physical page, and the swap location or file location if the page is
    not mapped.  The -u and -k options specify that the address is a user
    or kernel virtual address; -u and -k are not necessary on processors whose
    virtual addresses self-define themselves as user or kernel.  User addresses
    are translated with respect to the current context unless the -c option
    is used.  Kernel virtual addresses are translated using the swapper_pg_dir
    as the base page directory unless the -c option is used.

     -u                 The address is a user virtual address; only required
                        on processors with overlapping user and kernel virtual
                        address spaces.
     -k                 The address is a kernel virtual address; only required
                        on processors with overlapping user and kernel virtual
                        address spaces.
     -c [pid | taskp]   Translate the virtual address from the page directory
                        of the specified PID or hexadecimal task_struct pointer.
                        However, if this command is invoked from "foreach vtop",
                        the pid or taskp argument should NOT be entered; the
                        address will be translated using the page directory of
                        each task specified by "foreach".
     address            A hexadecimal user or kernel virtual address.

EXAMPLES
  Translate user virtual address 80b4000:

    crash> vtop 80b4000
    VIRTUAL   PHYSICAL
    80b4000   660f000

    PAGE DIRECTORY: c37f0000
      PGD: c37f0080 => e0d067
      PMD: c37f0080 => e0d067
      PTE: c0e0d2d0 => 660f067
     PAGE: 660f000

      PTE    PHYSICAL  FLAGS
    660f067   660f000  (PRESENT|RW|USER|ACCESSED|DIRTY)

      VMA      START      END      FLAGS  FILE
    c773daa0   80b4000   810c000    77

      PAGE    PHYSICAL   INODE     OFFSET  CNT FLAGS
    c0393258   660f000         0     17000  1  uptodate

  Translate kernel virtual address c806e000, first using swapper_pg_dir
  as the page directory base, and secondly, using the page table base
  of PID 1359:

    crash> vtop c806e000
    VIRTUAL   PHYSICAL
    c806e000  2216000

    PAGE DIRECTORY: c0101000
      PGD: c0101c80 => 94063
      PMD: c0101c80 => 94063
      PTE: c00941b8 => 2216063
     PAGE: 2216000

      PTE    PHYSICAL  FLAGS
    2216063   2216000  (PRESENT|RW|ACCESSED|DIRTY)

      PAGE    PHYSICAL   INODE     OFFSET  CNT FLAGS
    c02e9370   2216000         0         0  1

    crash> vtop -c 1359 c806e000
    VIRTUAL   PHYSICAL
    c806e000  2216000

    PAGE DIRECTORY: c5caf000
      PGD: c5cafc80 => 94063
      PMD: c5cafc80 => 94063
      PTE: c00941b8 => 2216063
     PAGE: 2216000
```

```
        PTE    PHYSICAL  FLAGS
      2216063   2216000  (PRESENT|RW|ACCESSED|DIRTY)

        PAGE    PHYSICAL   INODE     OFFSET  CNT FLAGS
      c02e9370   2216000         0          0  1

    Determine swap location of user virtual address 40104000:

      crash> vtop 40104000
      VIRTUAL   PHYSICAL
      40104000  (not mapped)

      PAGE DIRECTORY: c40d8000
        PGD: c40d8400 => 6bbe067
        PMD: c40d8400 => 6bbe067
        PTE: c6bbe410 => 58bc00

       PTE      SWAP     OFFSET
      58bc00  /dev/sda8   22716

        VMA      START      END     FLAGS  FILE
      c7200ae0  40104000  40b08000    73

      SWAP: /dev/sda8  OFFSET: 22716


NAME
  bt - backtrace

SYNOPSIS
  bt [-a|-c cpu(s)|-g|-r|-t|-T|-l|-e|-E|-f|-F|-o|-O|-v|-p] [-R ref] [-s [-x|d]]
     [-I ip] [-S sp] [-n idle] [pid | task]

DESCRIPTION
  Display a kernel stack backtrace.  If no arguments are given, the stack
  trace of the current context will be displayed.

       -a  displays the stack traces of the active task on each CPU.
           (only applicable to crash dumps)
       -A  same as -a, but also displays vector registers (S390X only).
  -n idle  filter the stack of idle tasks (x86_64, arm64).
           (only applicable to crash dumps)
       -p  display the stack trace of the panic task only.
           (only applicable to crash dumps)
   -c cpu  display the stack trace of the active task on one or more CPUs,
           which can be specified using the format "3", "1,8,9", "1-23",
           or "1,8,9-14". (only applicable to crash dumps)
       -g  displays the stack traces of all threads in the thread group of
           the target task; the thread group leader will be displayed first.
       -r  display raw stack data, consisting of a memory dump of the two
           pages of memory containing the task_union structure.
       -t  display all text symbols found from the last known stack location
           to the top of the stack. (helpful if the back trace fails)
       -T  display all text symbols found from just above the task_struct or
           thread_info to the top of the stack. (helpful if the back trace
           fails or the -t option starts too high in the process stack).
       -l  show file and line number of each stack trace text location.
       -e  search the stack for possible kernel and user mode exception frames.
       -E  search the IRQ stacks (x86, x86_64, arm64, and ppc64), and the
           exception stacks (x86_64) for possible exception frames; all other
           arguments except for -c will be ignored since this is not a context-
           sensitive operation.
       -f  display all stack data contained in a frame; this option can be
           used to determine the arguments passed to each function; on ia64,
           the argument register contents are dumped.
    -F[F]  similar to -f, except that the stack data is displayed symbolically
           when appropriate; if the stack data references a slab cache object,
           the name of the slab cache will be displayed in brackets; on ia64,
           the substitution is done to the argument register contents.  If -F
           is entered twice, and the stack data references a slab cache object,
           both the address and the name of the slab cache will be displayed
           in brackets.
```

```
        -v  check the kernel stack of all tasks for evidence of stack overflows.
            It does so by verifying the thread_info.task pointer, ensuring that
            the thread_info.cpu is a valid cpu number, and checking the end of
            the stack for the STACK_END_MAGIC value.
        -o  arm64: use optional backtrace method; not supported on Linux 4.14 or
            later kernels.
            x86: use old backtrace method, permissible only on kernels that were
            compiled without the -fomit-frame_pointer.
            x86_64: use old backtrace method, which dumps potentially stale
            kernel text return addresses found on the stack.
        -O  arm64: use optional backtrace method by default; subsequent usage
            of this option toggles the backtrace method.
            x86: use old backtrace method by default, permissible only on kernels
            that were compiled without the -fomit-frame_pointer; subsequent usage
            of this option toggles the backtrace method.
            x86_64: use old backtrace method by default; subsequent usage of this
            option toggles the backtrace method.
     -R ref  display stack trace only if there is a reference to this symbol
            or text address.
        -s  display the symbol name plus its offset.
        -x  when displaying a symbol offset with the -s option, override the
            default output format with hexadecimal format.
        -d  when displaying a symbol offset with the -s option, override the
            default output format with decimal format.
      -I ip  use ip as the starting text location.
      -S sp  use sp as the starting stack frame address.
        pid  displays the stack trace(s) of this pid.
      taskp  displays the stack trace the the task referenced by this hexadecimal
            task_struct pointer.

  Multiple pid and taskp arguments may be specified.

  Note that all examples below are for x86 only.  The output format will differ
  for other architectures.  x86 backtraces from kernels that were compiled
  with the --fomit-frame-pointer CFLAG occasionally will drop stack frames,
  or display a stale frame reference.  When in doubt as to the accuracy of a
  backtrace, the -t or -T options may help fill in the blanks.

EXAMPLES
  Display the stack trace of the active task(s) when the kernel panicked:

    crash> bt -a
    PID: 286    TASK: c0b3a000  CPU: 0   COMMAND: "in.rlogind"
    #0 [c0b3be90] crash_save_current_state at c011aed0
    #1 [c0b3bea4] panic at c011367c
    #2 [c0b3bee8] tulip_interrupt at c01bc820
    #3 [c0b3bf08] handle_IRQ_event at c010a551
    #4 [c0b3bf2c] do_8259A_IRQ at c010a319
    #5 [c0b3bf3c] do_IRQ at c010a653
    #6 [c0b3bfbc] ret_from_intr at c0109634
       EAX: 00000000  EBX: c0e68280  ECX: 00000000  EDX: 00000004  EBP: c0b3bfbc
       DS:  0018      ESI: 00000004  ES:  0018      EDI: c0e68284
       CS:  0010      EIP: c012f803  ERR: ffffff09  EFLAGS: 00000246
    #7 [c0b3bfbc] sys_select at c012f803
    #8 [c0b3bfc0] system_call at c0109598
       EAX: 0000008e  EBX: 00000004  ECX: bfffc9a0  EDX: 00000000
       DS:  002b      ESI: bfffc8a0  ES:  002b      EDI: 00000000
       SS:  002b      ESP: bfffc82c  EBP: bfffd224
       CS:  0023      EIP: 400d032e  ERR: 0000008e  EFLAGS: 00000246

  Display the stack trace of the active task(s) when the kernel panicked,
  and filter out the stack of the idle tasks:

    crash> bt -a -n idle
    ...
    PID: 0      TASK: ffff889ff8c35a00  CPU: 11  COMMAND: "swapper/11"

    PID: 0      TASK: ffff889ff8c3c380  CPU: 12  COMMAND: "swapper/12"

    PID: 150773  TASK: ffff889fe85a1680  CPU: 13  COMMAND: "bash"
    #0 [ffffc9000d35bcd0] machine_kexec at ffffffff8105a407
    #1 [ffffc9000d35bd28] __crash_kexec at ffffffff8113033d
    #2 [ffffc9000d35bdf0] panic at ffffffff81081930
```

```
    #3 [ffffc9000d35be70] sysrq_handle_crash at ffffffff814e38d1
    #4 [ffffc9000d35be78] __handle_sysrq.cold.12 at ffffffff814e4175
    #5 [ffffc9000d35bea8] write_sysrq_trigger at ffffffff814e404b
    #6 [ffffc9000d35beb8] proc_reg_write at ffffffff81330d86
    #7 [ffffc9000d35bed0] vfs_write at ffffffff812a72d5
    #8 [ffffc9000d35bf00] ksys_write at ffffffff812a7579
    #9 [ffffc9000d35bf38] do_syscall_64 at ffffffff81004259
       RIP: 00007fa7abcdc274  RSP: 00007fffa731f678  RFLAGS: 00000246
       RAX: ffffffffffffffda  RBX: 0000000000000002  RCX: 00007fa7abcdc274
       RDX: 0000000000000002  RSI: 0000563ca51ee6d0  RDI: 0000000000000001
       RBP: 0000563ca51ee6d0   R8: 000000000000000a   R9: 00007fa7abd6be80
       R10: 000000000000000a  R11: 0000000000000246  R12: 00007fa7abdad760
       R13: 0000000000000002  R14: 00007fa7abda8760  R15: 0000000000000002
       ORIG_RAX: 0000000000000001  CS: 0033  SS: 002b
    ...

Display the stack trace of the active task on CPU 0 and 1:

    crash> bt -c 0,1
    PID: 0      TASK: ffffffff81a8d020  CPU: 0   COMMAND: "swapper"
     #0 [ffff880002207e90] crash_nmi_callback at ffffffff8102fee6
     #1 [ffff880002207ea0] notifier_call_chain at ffffffff8152d525
     #2 [ffff880002207ee0] atomic_notifier_call_chain at ffffffff8152d58a
     #3 [ffff880002207ef0] notify_die at ffffffff810a155e
     #4 [ffff880002207f20] do_nmi at ffffffff8152b1eb
     #5 [ffff880002207f50] nmi at ffffffff8152aab0
        [exception RIP: native_safe_halt+0xb]
        RIP: ffffffff8103eacb  RSP: ffffffff81a01ea8  RFLAGS: 00000296
        RAX: 0000000000000000  RBX: 0000000000000000  RCX: 0000000000000000
        RDX: 0000000000000000  RSI: 0000000000000001  RDI: ffffffff81de5228
        RBP: ffffffff81a01ea8   R8: 0000000000000000   R9: 0000000000000000
        R10: 0012099429a6bea3  R11: 0000000000000000  R12: ffffffff81c066c0
        R13: 0000000000000000  R14: ffffffffffffffff  R15: ffffffff81de1000
        ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
    --- <NMI exception stack> ---
     #6 [ffffffff81a01ea8] native_safe_halt at ffffffff8103eacb
     #7 [ffffffff81a01eb0] default_idle at ffffffff810167bd
     #8 [ffffffff81a01ed0] cpu_idle at ffffffff81009fc6

    PID: 38     TASK: ffff88003eaae040  CPU: 1   COMMAND: "khungtaskd"
     #0 [ffff88003ad97ce8] machine_kexec at ffffffff81038f3b
     #1 [ffff88003ad97d48] crash_kexec at ffffffff810c5da2
     #2 [ffff88003ad97e18] panic at ffffffff8152721a
     #3 [ffff88003ad97e98] watchdog at ffffffff810e6346
     #4 [ffff88003ad97ee8] kthread at ffffffff8109af06
     #5 [ffff88003ad97f48] kernel_thread at ffffffff8100c20a

Display the stack traces of task f2814000 and PID 1592:

    crash> bt f2814000 1592
    PID: 1018   TASK: f2814000  CPU: 1   COMMAND: "java"
     #0 [f2815db4] schedule at c011af85
     #1 [f2815de4] __down at c010600f
     #2 [f2815e14] __down_failed at c01061b3
     #3 [f2815e24] stext_lock (via drain_cpu_caches) at c025fa55
     #4 [f2815ec8] kmem_cache_shrink_nr at c013a53e
     #5 [f2815ed8] do_try_to_free_pages at c013f402
     #6 [f2815f04] try_to_free_pages at c013f8d2
     #7 [f2815f1c] _wrapped_alloc_pages at c01406bd
     #8 [f2815f40] __alloc_pages at c014079d
     #9 [f2815f60] __get_free_pages at c014083e
    #10 [f2815f68] do_fork at c011cebb
    #11 [f2815fa4] sys_clone at c0105ceb
    #12 [f2815fc0] system_call at c010740c
        EAX: 00000078  EBX: 00000f21  ECX: bc1ffbd8  EDX: bc1ffbe0
        DS:  002b      ESI: 00000000  ES:  002b      EDI: bc1ffd04
        SS:  002b      ESP: 0807316c  EBP: 080731bc
        CS:  0023      EIP: 4012881e  ERR: 00000078  EFLAGS: 00000296

    PID: 1592   TASK: c0cec000  CPU: 3   COMMAND: "httpd"
     #0 [c0ceded4] schedule at c011af85
     #1 [c0cedf04] pipe_wait at c0153083
     #2 [c0cedf58] pipe_read at c015317f
```

```
 #3 [c0cedf7c] sys_read at c0148be6
 #4 [c0cedfc0] system_call at c010740c
    EAX: 00000003  EBX: 00000004  ECX: bffed4a3  EDX: 00000001
    DS:  002b      ESI: 00000001  ES:  002b      EDI: bffed4a3
    SS:  002b      ESP: bffed458  EBP: bffed488
    CS:  0023      EIP: 4024f1d4  ERR: 00000003  EFLAGS: 00000286
```

In order to examine each stack frame's contents use the bt -f option. From the extra frame data that is displayed, the arguments passed to each function can be determined. Re-examining the PID 1592 trace above:

```
crash> bt -f 1592
PID: 1592   TASK: c0cec000  CPU: 3   COMMAND: "httpd"
 #0 [c0ceded4] schedule at c011af85
    [RA: c0153088  SP: c0ceded4  FP: c0cedf04  SIZE: 52]
    c0ceded4: c0cedf00  c0cec000  ce1a6000  00000003
    c0cedee4: c0cec000  f26152c0  cfafc8c0  c0cec000
    c0cedef4: ef70a0a0  c0cec000  c0cedf28  c0cedf54
    c0cedf04: c0153088
 #1 [c0cedf04] pipe_wait at c0153083
    [RA: c0153184  SP: c0cedf08  FP: c0cedf58  SIZE: 84]
    c0cedf08: 00000000  c0cec000  00000000  00000000
    c0cedf18: 00000000  c0a41fa0  c011d38b  c0394120
    c0cedf28: 00000000  c0cec000  ceeebf30  ce4adf30
    c0cedf38: 00000000  d4b60ce0  00000000  c0cedf58
    c0cedf48: e204f820  ef70a040  00000001  c0cedf78
    c0cedf58: c0153184
 #2 [c0cedf58] pipe_read at c015317f
    [RA: c0148be8  SP: c0cedf5c  FP: c0cedf7c  SIZE: 36]
    c0cedf5c: ef70a040  c0cec000  00000000  00000000
    c0cedf6c: 00000001  f27ae680  ffffffea  c0cedfbc
    c0cedf7c: c0148be8
 #3 [c0cedf7c] sys_read at c0148be6
    [RA: c0107413  SP: c0cedf80  FP: c0cedfc0  SIZE: 68]
    c0cedf80: f27ae680  bffed4a3  00000001  f27ae6a0
    c0cedf90: 40160370  24000000  4019ba28  00000000
    c0cedfa0: 00000000  fffffffe  bffba207  fffffffe
    c0cedfb0: c0cec000  00000001  bffed4a3  bffed488
    c0cedfc0: c0107413
 #4 [c0cedfc0] system_call at c010740c
    EAX: 00000003  EBX: 00000004  ECX: bffed4a3  EDX: 00000001
    DS:  002b      ESI: 00000001  ES:  002b      EDI: bffed4a3
    SS:  002b      ESP: bffed458  EBP: bffed488
    CS:  0023      EIP: 4024f1d4  ERR: 00000003  EFLAGS: 00000286
    [RA: 4024f1d4  SP: c0cedfc4  FP: c0cedffc  SIZE: 60]
    c0cedfc4: 00000004  bffed4a3  00000001  00000001
    c0cedfd4: bffed4a3  bffed488  00000003  0000002b
    c0cedfe4: 0000002b  00000003  4024f1d4  00000023
    c0cedff4: 00000286  bffed458  0000002b
```

Typically the arguments passed to a function will be the last values that were pushed onto the stack by the next higher-numbered function, i.e., the lowest stack addresses in the frame above the called function's stack frame. That can be verified by disassembling the calling function. For example, the arguments passed from sys_read() to pipe_read() above are the file pointer, the user buffer address, the count, and a pointer to the file structure's f_pos field. Looking at the frame #3 data for sys_read(), the last four items pushed onto the stack (lowest addresses) are f27ae680, bffed4a3, 00000001, and f27ae6a0 -- which are the 4 arguments above, in that order. Note that the first (highest address) stack content in frame #2 data for pipe_read() is c0148be8, which is the return address back to sys_read().

Dump the text symbols found in the current context's stack:

```
crash> bt -t
PID: 1357   TASK: c1aa0000  CPU: 0   COMMAND: "lockd"
      START: schedule at c01190e0
  [c1aa1f28] dput at c0157dbc
  [c1aa1f4c] schedule_timeout at c0124cd4
  [c1aa1f78] svc_recv at cb22c4d8 [sunrpc]
  [c1aa1f98] put_files_struct at c011eb21
  [c1aa1fcc] nlmclnt_proc at cb237bef [lockd]
```

```
    [c1aa1ff0] kernel_thread at c0105826
    [c1aa1ff8] nlmclnt_proc at cb237a60 [lockd]
```

Search the current stack for possible exception frames:

```
crash> bt -e
PID: 286    TASK: c0b3a000  CPU: 0   COMMAND: "in.rlogind"

 KERNEL-MODE EXCEPTION FRAME AT c0b3bf44:
   EAX: 00000000  EBX: c0e68280  ECX: 00000000  EDX: 00000004  EBP: c0b3bfbc
   DS:  0018      ESI: 00000004  ES:  0018      EDI: c0e68284
   CS:  0010      EIP: c012f803  ERR: ffffff09  EFLAGS: 00000246

 USER-MODE EXCEPTION FRAME AT c0b3bfc4:
   EAX: 0000008e  EBX: 00000004  ECX: bfffc9a0  EDX: 00000000
   DS:  002b      ESI: bfffc8a0  ES:  002b      EDI: 00000000
   SS:  002b      ESP: bfffc82c  EBP: bfffd224
   CS:  0023      EIP: 400d032e  ERR: 0000008e  EFLAGS: 00000246
```

Display the back trace from a dumpfile that resulted from the execution of the crash utility's "sys -panic" command:

```
 crash> bt
 PID: 12523  TASK: c610c000  CPU: 0   COMMAND: "crash"
  #0 [c610de64] die at c01076ec
  #1 [c610de74] do_invalid_op at c01079bc
  #2 [c610df2c] error_code (via invalid_op) at c0107256
     EAX: 0000001d  EBX: c024a4c0  ECX: c02f13c4  EDX: 000026f6  EBP: c610c000
     DS:  0018      ESI: 401de2e0  ES:  0018      EDI: c610c000
     CS:  0010      EIP: c011bbb4  ERR: ffffffff  EFLAGS: 00010296
  #3 [c610df68] panic at c011bbb4
  #4 [c610df78] do_exit at c011f1fe
  #5 [c610dfc0] system_call at c0107154
     EAX: 00000001  EBX: 00000000  ECX: 00001000  EDX: 401df154
     DS:  002b      ESI: 401de2e0  ES:  002b      EDI: 00000000
     SS:  002b      ESP: bffebf0c  EBP: bffebf38
     CS:  0023      EIP: 40163afd  ERR: 00000001  EFLAGS: 00000246
```

Display the back trace from a dumpfile that resulted from an attempt to insmod the sample "crash.c" kernel module that comes as part of the Red Hat netdump package:

```
 crash> bt
 PID: 1696   TASK: c74de000  CPU: 0   COMMAND: "insmod"
  #0 [c74dfdcc] die at c01076ec
  #1 [c74dfddc] do_page_fault at c0117bbc
  #2 [c74dfee0] error_code (via page_fault) at c0107256
     EAX: 00000013  EBX: cb297000  ECX: 00000000  EDX: c5962000  EBP: c74dff28
     DS:  0018      ESI: 00000000  ES:  0018      EDI: 00000000
     CS:  0010      EIP: cb297076  ERR: ffffffff  EFLAGS: 00010282
  #3 [c74dff1c] crash_init at cb297076 [crash]
  #4 [c74dff2c] sys_init_module at c011d233
  #5 [c74dffc0] system_call at c0107154
     EAX: 00000080  EBX: 08060528  ECX: 08076450  EDX: 0000000a
     DS:  002b      ESI: 0804b305  ES:  002b      EDI: 08074ed0
     SS:  002b      ESP: bffe9a90  EBP: bffe9ac8
     CS:  0023      EIP: 4012066e  ERR: 00000080  EFLAGS: 00000246
```

Display the symbol name plus its offset in each frame, overriding the current output format with hexadecimal:

```
crash> bt -sx
PID: 1499   TASK: ffff88006af43cc0  CPU: 2   COMMAND: "su"
 #0 [ffff8800664a1c90] machine_kexec+0x167 at ffffffff810327b7
 #1 [ffff8800664a1ce0] crash_kexec+0x60 at ffffffff810a9ec0
 #2 [ffff8800664a1db0] oops_end+0xb0 at ffffffff81504160
 #3 [ffff8800664a1dd0] general_protection+0x25 at ffffffff81503435
    [exception RIP: kmem_cache_alloc+120]
    RIP: ffffffff8113cf88  RSP: ffff8800664a1e88  RFLAGS: 00010086
    RAX: 0000000000000000  RBX: ff88006ef56840ff  RCX: ffffffff8114e9e4
    RDX: 0000000000000000  RSI: 00000000000080d0  RDI: ffffffff81796020
    RBP: ffffffff81796020   R8: ffff88000a3137a0   R9: 0000000000000000
    R10: ffff88007ac97300  R11: 0000000000000400  R12: 00000000000080d0
```

```
          R13: 0000000000000292  R14: 00000000000080d0  R15: 00000000000000c0
          ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
      #4 [ffff8800664a1ed0] get_empty_filp+0x74 at ffffffff8114e9e4
      #5 [ffff8800664a1ef0] sock_alloc_fd+0x23 at ffffffff8142f553
      #6 [ffff8800664a1f10] sock_map_fd+0x23 at ffffffff8142f693
      #7 [ffff8800664a1f50] sys_socket+0x43 at ffffffff814302a3
      #8 [ffff8800664a1f80] system_call_fastpath+0x16 at ffffffff81013042
          RIP: 00007f5720b368e7  RSP: 00007fff52b629a8  RFLAGS: 00010206
          RAX: 0000000000000029  RBX: ffffffff81013042  RCX: 0000000000000000
          RDX: 0000000000000009  RSI: 0000000000000003  RDI: 0000000000000010
          RBP: 000000000066f320   R8: 0000000000000001   R9: 0000000000000000
          R10: 0000000000000000  R11: 0000000000000202  R12: ffff88007ac97300
          R13: 0000000000000000  R14: 00007f571e104a80  R15: 00007f571e305048
          ORIG_RAX: 0000000000000029  CS: 0033  SS: 002b

  The following three examples show the difference in the display of
  the same stack frame's contents using -f, -F, and -FF:

    crash> bt -f
    ...
     #4 [ffff810072b47f10] vfs_write at ffffffff800789d8
        ffff810072b47f18: ffff81007e020380 ffff81007e2c2880
        ffff810072b47f28: 0000000000000002 fffffffffffffff7
        ffff810072b47f38: 00002b141825d000 ffffffff80078f75
     #5 [ffff810072b47f40] sys_write at ffffffff80078f75
    ...
    crash> bt -F
    ...
     #4 [ffff810072b47f10] vfs_write at ffffffff800789d8
        ffff810072b47f18: [files_cache]    [filp]
        ffff810072b47f28: 0000000000000002 fffffffffffffff7
        ffff810072b47f38: 00002b141825d000 sys_write+69
     #5 [ffff810072b47f40] sys_write at ffffffff80078f75
    ...
    crash> bt -FF
    ...
     #4 [ffff810072b47f10] vfs_write at ffffffff800789d8
        ffff810072b47f18: [ffff81007e020380:files_cache] [ffff81007e2c2880:filp]
        ffff810072b47f28: 0000000000000002 fffffffffffffff7
        ffff810072b47f38: 00002b141825d000 sys_write+69
     #5 [ffff810072b47f40] sys_write at ffffffff80078f75
    ...

  Check the kernel stack of all tasks for evidence of a stack overflow:

    crash> bt -v
    PID: 5823   TASK: ffff88102aae0040  CPU: 1   COMMAND: "flush-253:0"
    possible stack overflow: thread_info.task: 102efb5adc0 != ffff88102aae0040
    possible stack overflow: 40ffffffff != STACK_END_MAGIC


NAME
  help - get help

SYNOPSIS
  help [command | all] [-<option>]

DESCRIPTION
  When entered with no argument, a list of all currently available crash
  commands is listed.  If a name of a crash command is entered, a man-like
  page for the command is displayed.  If "all" is entered, help pages
  for all commands will be displayed.  If neither of the above is entered,
  the argument string will be passed on to the gdb help command.

  A number of internal debug, statistical, and other dumpfile related
  data is available with the following options:

    -a - alias data
    -b - shared buffer data
    -B - build data
    -c - numargs cache
    -d - device table
    -D - dumpfile contents/statistics
```

```
      -e - extension table data
      -f - filesys table
      -g - gdb data
      -h - hash_table data
      -H - hash_table data (verbose)
      -k - kernel_table
      -K - kernel_table (verbose)
      -L - LKCD page cache environment
      -M <num> machine specific
      -m - machdep_table
      -N - net_table
      -n - dumpfile contents/statistics
      -o - offset_table and size_table
      -p - program_context
      -r - dump registers from dumpfile header
      -s - symbol table data
      -t - task_table
      -T - task_table plus context_array
      -v - vm_table
      -V - vm_table (verbose)
      -z - help options


NAME
  ps - display process status information

SYNOPSIS
  ps [-k|-u|-G|-y policy] [-s] [-p|-c|-t|-[l|m][-C cpu]|-a|-g|-r|-S|-A]
     [pid | task | command] ...

DESCRIPTION
  This command displays process status for selected, or all, processes
  in the system.  If no arguments are entered, the process data is
  is displayed for all processes.  Specific processes may be selected
  by using the following identifier formats:

       pid  a process PID.
      task  a hexadecimal task_struct pointer.
   command  a command name.  If a command name is made up of letters that
            are all numerical values, precede the name string with a "\".
            If the command string is enclosed within "'" characters, then
            the encompassed string must be a POSIX extended regular expression
            that will be used to match task names.

  The process list may be further restricted by the following options:

        -k  restrict the output to only kernel threads.
        -u  restrict the output to only user tasks.
        -G  display only the thread group leader in a thread group.
 -y policy  restrict the output to tasks having a specified scheduling policy
            expressed by its integer value or by its (case-insensitive) name;
            multiple policies may be entered in a comma-separated list:
              0 or NORMAL
              1 or FIFO
              2 or RR
              3 or BATCH
              4 or ISO
              5 or IDLE
              6 or DEADLINE

  The process identifier types may be mixed.  For each task, the following
  items are displayed:

    1. the process PID.
    2. the parent process PID.
    3. the CPU number that the task ran on last.
    4. the task_struct address or the kernel stack pointer of the process.
       (see -s option below)
    5. the task state (RU, IN, UN, ZO, ST, TR, DE, SW, WA, PA, ID, NE).
    6. the percentage of physical memory being used by this task.
    7. the virtual address size of this task in kilobytes.
    8. the resident set size of this task in kilobytes.
    9. the command name.
```

The default output shows the task_struct address of each process under a column titled "TASK". This can be changed to show the kernel stack pointer under a column titled "KSTACKP".

-s replace the TASK column with the KSTACKP column.

On SMP machines, the active task on each CPU will be highlighted by an angle bracket (">") preceding its information. If the crash variable "offline" is set to "hide", the active task on an offline CPU will be highlighted by a "-" preceding its information.

Alternatively, information regarding parent-child relationships, per-task time usage data, argument/environment data, thread groups, or resource limits may be displayed:

-p display the parental hierarchy of selected, or all, tasks.
-c display the children of selected, or all, tasks.
-t display the task run time, start time, and cumulative user and system times.
-l display the task's last-run timestamp value, using either the task_struct's last_run value, the task_struct's timestamp value or the task_struct's sched_entity last_arrival value, whichever applies, of selected, or all, tasks; the list is sorted with the most recently-run task (with the largest timestamp) shown first, followed by the task's current state.
-m similar to -l, but the timestamp value is translated into days, hours, minutes, seconds, and milliseconds since the task was last run on a cpu.
-C cpus only usable with the -l or -m options, dump the timestamp data in per-cpu blocks, where the cpu[s] can be specified as "1,3,5", "1-3", "1,3,5-7,10", "all", or "a" (shortcut for "all").
-a display the command line arguments and environment strings of selected, or all, user-mode tasks.
-g display tasks by thread group, of selected, or all, tasks.
-r display resource limits (rlimits) of selected, or all, tasks.
-S display a summary consisting of the number of tasks in a task state.
-A display only the active task on each cpu.

EXAMPLES

Show the process status of all current tasks:

```
crash> ps
   PID    PPID  CPU   TASK    ST  %MEM   VSZ   RSS  COMM
>     0      0   3  c024c000  RU   0.0     0     0  [swapper]
>     0      0   0  c0dce000  RU   0.0     0     0  [swapper]
      0      0   1  c0fa8000  RU   0.0     0     0  [swapper]
>     0      0   2  c009a000  RU   0.0     0     0  [swapper]
      1      0   1  c0098000  IN   0.0  1096   476  init
      2      1   1  c0090000  IN   0.0     0     0  [kflushd]
      3      1   1  c000e000  IN   0.0     0     0  [kpiod]
      4      1   3  c000c000  IN   0.0     0     0  [kswapd]
      5      1   1  c0008000  IN   0.0     0     0  [mdrecoveryd]
    253      1   2  fbc4c000  IN   0.0  1088   376  portmap
    268      1   2  fbc82000  IN   0.1  1232   504  ypbind
    274    268   2  fa984000  IN   0.1  1260   556  ypbind
    321      1   1  fabf6000  IN   0.1  1264   608  syslogd
    332      1   1  fa9be000  RU   0.1  1364   736  klogd
    346      1   2  fae88000  IN   0.0  1112   472  atd
    360      1   2  faeb2000  IN   0.1  1284   592  crond
    378      1   2  fafd6000  IN   0.1  1236   560  inetd
    392      1   0  fb710000  IN   0.1  2264  1468  named
    406      1   3  fb768000  IN   0.1  1284   560  lpd
    423      1   1  fb8ac000  IN   0.1  1128   528  rpc.statd
    434      1   2  fb75a000  IN   0.0  1072   376  rpc.rquotad
    445      1   2  fb4a4000  IN   0.0  1132   456  rpc.mountd
    460      1   1  fa938000  IN   0.0     0     0  [nfsd]
    461      1   1  faa86000  IN   0.0     0     0  [nfsd]
    462      1   0  fac48000  IN   0.0     0     0  [nfsd]
    463      1   0  fb4ca000  IN   0.0     0     0  [nfsd]
    464      1   0  fb4c8000  IN   0.0     0     0  [nfsd]
    465      1   2  fba6e000  IN   0.0     0     0  [nfsd]
    466      1   1  fba6c000  IN   0.0     0     0  [nfsd]
```

```
          467      1   2  fac04000  IN   0.0     0     0  [nfsd]
          468    461   2  fa93a000  IN   0.0     0     0  [lockd]
          469    468   2  fa93e000  IN   0.0     0     0  [rpciod]
          486      1   0  fab54000  IN   0.1  1596   880  amd
          523      1   2  fa84e000  IN   0.1  1884  1128  sendmail
          538      1   0  fa82c000  IN   0.0  1112   416  gpm
          552      1   3  fa70a000  IN   0.1  2384  1220  httpd
          556    552   3  fa776000  IN   0.1  2572  1352  httpd
          557    552   2  faba4000  IN   0.1  2572  1352  httpd
          558    552   1  fa802000  IN   0.1  2572  1352  httpd
          559    552   3  fa6ee000  IN   0.1  2572  1352  httpd
          560    552   3  fa700000  IN   0.1  2572  1352  httpd
          561    552   0  fa6f0000  IN   0.1  2572  1352  httpd
          562    552   3  fa6ea000  IN   0.1  2572  1352  httpd
          563    552   0  fa67c000  IN   0.1  2572  1352  httpd
          564    552   3  fa674000  IN   0.1  2572  1352  httpd
          565    552   3  fa66a000  IN   0.1  2572  1352  httpd
          582      1   2  fa402000  IN   0.2  2968  1916  xfs
          633      1   2  fa1ec000  IN   0.2  5512  2248  innd
          636      1   3  fa088000  IN   0.1  2536   804  actived
          676      1   0  fa840000  IN   0.0  1060   384  mingetty
          677      1   1  fa590000  IN   0.0  1060   384  mingetty
          678      1   2  fa3b8000  IN   0.0  1060   384  mingetty
          679      1   0  fa5b8000  IN   0.0  1060   384  mingetty
          680      1   1  fa3a4000  IN   0.0  1060   384  mingetty
          681      1   2  fa30a000  IN   0.0  1060   384  mingetty
          683      1   3  fa5d8000  IN   0.0  1052   280  update
          686    378   1  fa3aa000  IN   0.1  2320  1136  in.rlogind
          687    686   2  f9e52000  IN   0.1  2136  1000  login
          688    687   0  f9dec000  IN   0.1  1732   976  bash
      >   700    688   1  f9d62000  RU   0.0  1048   256  gen12
```

Display the parental hierarchy of the "crash" process on a live system:

```
  crash> ps -p 4249
  PID: 0      TASK: c0252000  CPU: 0   COMMAND: "swapper"
   PID: 1      TASK: c009a000  CPU: 1   COMMAND: "init"
    PID: 632    TASK: c73b6000  CPU: 1   COMMAND: "prefdm"
     PID: 637    TASK: c5a4a000  CPU: 1   COMMAND: "prefdm"
      PID: 649    TASK: c179a000  CPU: 0   COMMAND: "kwm"
       PID: 683    TASK: c1164000  CPU: 0   COMMAND: "kfm"
        PID: 1186   TASK: c165a000  CPU: 0   COMMAND: "xterm"
         PID: 1188   TASK: c705e000  CPU: 1   COMMAND: "bash"
          PID: 4249   TASK: c6b9a000  CPU: 0   COMMAND: "crash"
```

Display all children of the "kwm" window manager:

```
  crash> ps -c kwm
    PID: 649    TASK: c179a000  CPU: 0   COMMAND: "kwm"
    PID: 682    TASK: c2d58000  CPU: 1   COMMAND: "kwmsound"
    PID: 683    TASK: c1164000  CPU: 1   COMMAND: "kfm"
    PID: 685    TASK: c053c000  CPU: 0   COMMAND: "krootwm"
    PID: 686    TASK: c13fa000  CPU: 0   COMMAND: "kpanel"
    PID: 687    TASK: c13f0000  CPU: 1   COMMAND: "kbgndwm"
```

Display all threads in a firefox session:

```
  crash> ps firefox
     PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
    21273  21256   6  ffff81003ec15080  IN  46.3 1138276 484364  firefox
    21276  21256   6  ffff81003f49e7e0  IN  46.3 1138276 484364  firefox
    21280  21256   0  ffff81003ec1d7e0  IN  46.3 1138276 484364  firefox
    21286  21256   6  ffff81000b0d1820  IN  46.3 1138276 484364  firefox
    21287  21256   2  ffff81000b0d10c0  IN  46.3 1138276 484364  firefox
    26975  21256   5  ffff81003b5c1820  IN  46.3 1138276 484364  firefox
    26976  21256   5  ffff810023232820  IN  46.3 1138276 484364  firefox
    26977  21256   4  ffff810021a11820  IN  46.3 1138276 484364  firefox
    26978  21256   5  ffff810003159040  IN  46.3 1138276 484364  firefox
    26979  21256   5  ffff81003a058820  IN  46.3 1138276 484364  firefox
```

Display only the thread group leader in the firefox session:

```
  crash> ps -G firefox
```

```
      PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
     21273  21256   0  ffff81003ec15080  IN  46.3 1138276 484364  firefox
```

Show the time usage data for pid 10318:

```
  crash> ps -t 10318
  PID: 10318  TASK: f7b85550  CPU: 5   COMMAND: "bash"
      RUN TIME: 1 days, 01:35:32
    START TIME: 5209
         UTIME: 95
         STIME: 57
```

Show the process status of PID 1, task f9dec000, and all nfsd tasks:

```
  crash> ps 1 f9dec000 nfsd
     PID    PPID  CPU   TASK    ST  %MEM   VSZ   RSS  COMM
       1      0   1  c0098000  IN   0.0  1096   476  init
     688    687   0  f9dec000  IN   0.1  1732   976  bash
     460      1   1  fa938000  IN   0.0     0     0  [nfsd]
     461      1   1  faa86000  IN   0.0     0     0  [nfsd]
     462      1   0  fac48000  IN   0.0     0     0  [nfsd]
     463      1   0  fb4ca000  IN   0.0     0     0  [nfsd]
     464      1   0  fb4c8000  IN   0.0     0     0  [nfsd]
     465      1   2  fba6e000  IN   0.0     0     0  [nfsd]
     466      1   1  fba6c000  IN   0.0     0     0  [nfsd]
     467      1   2  fac04000  IN   0.0     0     0  [nfsd]
```

Show all kernel threads:

```
  crash> ps -k
     PID    PPID  CPU   TASK    ST  %MEM   VSZ   RSS  COMM
       0      0   1  c0fac000  RU   0.0     0     0  [swapper]
       0      0   0  c0252000  RU   0.0     0     0  [swapper]
       2      1   1  c0fa0000  IN   0.0     0     0  [kflushd]
       3      1   1  c03de000  IN   0.0     0     0  [kpiod]
       4      1   1  c03dc000  IN   0.0     0     0  [kswapd]
       5      1   0  c0092000  IN   0.0     0     0  [mdrecoveryd]
     336      1   0  c4a9a000  IN   0.0     0     0  [rpciod]
     337      1   0  c4830000  IN   0.0     0     0  [lockd]
     487      1   1  c4ba6000  IN   0.0     0     0  [nfsd]
     488      1   0  c18c6000  IN   0.0     0     0  [nfsd]
     489      1   0  c0cac000  IN   0.0     0     0  [nfsd]
     490      1   0  c056a000  IN   0.0     0     0  [nfsd]
     491      1   0  c0860000  IN   0.0     0     0  [nfsd]
     492      1   1  c0254000  IN   0.0     0     0  [nfsd]
     493      1   0  c0a86000  IN   0.0     0     0  [nfsd]
     494      1   0  c0968000  IN   0.0     0     0  [nfsd]
```

Display a summary consisting of the number of tasks in a task state:

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

Display only the active task, on each cpu:

```
  crash> ps -A
      PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
   >    10      2   1  ffff880212969710  IN   0.0       0      0   [migration/1]
   >     0      0   3  ffff884026d43520  RU   0.0       0      0   [swapper]
   >  6582      1   2  ffff880f49c52040  RU   0.0  42202472  33368  oracle
   >  9497      1   0  ffff880549ec2ab0  RU   0.0  42314692 138664  oracle
```

Show all tasks sorted by their task_struct's last_run, timestamp, or sched_entity last_arrival timestamp value, whichever applies:

```
  crash> ps -l
  [20811245123] [IN] PID: 37    TASK: f7153030  CPU: 2  COMMAND: "events/2"
  [20811229959] [IN] PID: 1756  TASK: f2a5a570  CPU: 2  COMMAND: "ntpd"
  [20800696644] [IN] PID: 1456  TASK: f2b1f030  CPU: 4  COMMAND: "irqbalance"
  [20617047229] [IN] PID: 2324  TASK: f57f9570  CPU: 5  COMMAND: "flush-253:0"
```

```
    [20617029209] [IN] PID: 49    TASK: f7167030  CPU: 4  COMMAND: "bdi-default"
    [20438025365] [IN] PID: 345   TASK: f55c7ab0  CPU: 3  COMMAND: "mpt_poll_0"
    [20103026046] [IN] PID: 728   TASK: f72ba570  CPU: 3  COMMAND: "edac-poller"
    [20000189409] [IN] PID: 35    TASK: f7153ab0  CPU: 0  COMMAND: "events/0"
    [20000179905] [IN] PID: 48    TASK: f7167570  CPU: 0  COMMAND: "sync_supers"
    [19997120354] [IN] PID: 36    TASK: f7153570  CPU: 1  COMMAND: "events/1"
    [19991059209] [IN] PID: 38    TASK: f715fab0  CPU: 3  COMMAND: "events/3"
    [19988091608] [IN] PID: 39    TASK: f715f570  CPU: 4  COMMAND: "events/4"
    [19985076530] [IN] PID: 40    TASK: f715f030  CPU: 5  COMMAND: "events/5"
    [19982019106] [IN] PID: 41    TASK: f7161ab0  CPU: 6  COMMAND: "events/6"
    [19982016294] [IN] PID: 29    TASK: f7109ab0  CPU: 6  COMMAND: "ksoftirqd/6"
    [19838402345] [RU] PID: 2331  TASK: f297f570  CPU: 7  COMMAND: "bash"
    [19837129436] [IN] PID: 2326  TASK: f2ad5030  CPU: 6  COMMAND: "sshd"
    [19289476417] [IN] PID: 1772  TASK: f5665570  CPU: 5  COMMAND: "sendmail"
    ...
```

Show the most-recently run tasks on cpu 0 using both the -l and the -m options:

```
  crash> ps -m -C0
  CPU: 0
  [ 0 00:00:00.003] [RU] PID: 1205 TASK: dee03f20 CPU: 0 COMMAND: "insmod"
  [ 0 00:00:00.006] [RU] PID: 770  TASK: df9e9940 CPU: 0 COMMAND: "rsyslogd"
  [ 0 00:00:00.009] [IN] PID: 603  TASK: df9bcbc0 CPU: 0 COMMAND: "udevd"
  [ 0 00:00:00.010] [IN] PID: 348  TASK: df9ecbc0 CPU: 0 COMMAND: "udevd"
  [ 0 00:00:00.013] [IN] PID: 934  TASK: df9171a0 CPU: 0 COMMAND: "hald"
  [ 0 00:00:00.023] [IN] PID: 6    TASK: df443f20 CPU: 0 COMMAND: "events/0"
  [ 0 00:00:00.029] [IN] PID: 15   TASK: df46b280 CPU: 0 COMMAND: "kblockd/0"
  [ 0 00:00:00.101] [IN] PID: 1168 TASK: dee01940 CPU: 0 COMMAND: "bash"
  [ 0 00:00:01.404] [IN] PID: 272  TASK: dfa48ca0 CPU: 0 COMMAND: "flush-8:0"
  ...

  crash> ps -l -C0
  CPU: 0
  [137146164748] [RU] PID: 1205 TASK: dee03f20 CPU: 0 COMMAND: "insmod"
  [137142534372] [RU] PID: 770  TASK: df9e9940 CPU: 0 COMMAND: "rsyslogd"
  [137140168469] [IN] PID: 603  TASK: df9bcbc0 CPU: 0 COMMAND: "udevd"
  [137138826427] [IN] PID: 348  TASK: df9ecbc0 CPU: 0 COMMAND: "udevd"
  [137135214599] [IN] PID: 934  TASK: df9171a0 CPU: 0 COMMAND: "hald"
  [137125651275] [IN] PID: 6    TASK: df443f20 CPU: 0 COMMAND: "events/0"
  [137119564815] [IN] PID: 15   TASK: df46b280 CPU: 0 COMMAND: "kblockd/0"
  [137047715027] [IN] PID: 1168 TASK: dee01940 CPU: 0 COMMAND: "bash"
  [135744209052] [IN] PID: 272  TASK: dfa48ca0 CPU: 0 COMMAND: "flush-8:0"
  ...
```

Show the kernel stack pointer of each user task:

```
  crash> ps -us
     PID    PPID  CPU  KSTACKP  ST  %MEM   VSZ   RSS  COMM
       1       0   0  c009bedc  IN   0.0  1096    52  init
     239       1   0  c15e7ed8  IN   0.2  1332   224  pump
     280       1   1  c7cbdedc  IN   0.2  1092   208  portmap
     295       1   0  c7481edc  IN   0.0  1232     0  ypbind
     301     295   0  c7c7bf28  IN   0.1  1260   124  ypbind
     376       1   1  c5053f28  IN   0.0  1316    40  automount
     381       1   0  c34ddf28  IN   0.2  1316   224  automount
     391       1   1  c2777f28  IN   0.2  1316   224  automount
  ...
```

Display the argument and environment data for the automount task:

```
  crash> ps -a automount
  PID: 3948   TASK: f722ee30  CPU: 0   COMMAND: "automount"
  ARG: /usr/sbin/automount --timeout=60 /net program /etc/auto.net
  ENV: SELINUX_INIT=YES
       CONSOLE=/dev/console
       TERM=linux
       INIT_VERSION=sysvinit-2.85
       PATH=/sbin:/usr/sbin:/bin:/usr/bin
       LC_MESSAGES=en_US
       RUNLEVEL=3
       runlevel=3
       PWD=/
```

```
          LANG=ja_JP.UTF-8
          PREVLEVEL=N
          previous=N
          HOME=/
          SHLVL=2
          _=/usr/sbin/automount
```

Display the tasks in the thread group containing task c20ab0b0:

```
  crash> ps -g c20ab0b0
  PID: 6425   TASK: f72f50b0  CPU: 0   COMMAND: "firefox-bin"
    PID: 6516   TASK: f71bf1b0  CPU: 0   COMMAND: "firefox-bin"
    PID: 6518   TASK: d394b930  CPU: 0   COMMAND: "firefox-bin"
    PID: 6520   TASK: c20aa030  CPU: 0   COMMAND: "firefox-bin"
    PID: 6523   TASK: c20ab0b0  CPU: 0   COMMAND: "firefox-bin"
    PID: 6614   TASK: f1f181b0  CPU: 0   COMMAND: "firefox-bin"
```

Display the tasks in the thread group for each instance of the program named "multi-thread":

```
  crash> ps -g multi-thread
  PID: 2522   TASK: 1003f0dc7f0       CPU: 1   COMMAND: "multi-thread"
    PID: 2523   TASK: 10037b13030       CPU: 1   COMMAND: "multi-thread"
    PID: 2524   TASK: 1003e064030       CPU: 1   COMMAND: "multi-thread"
    PID: 2525   TASK: 1003e13a7f0       CPU: 1   COMMAND: "multi-thread"

  PID: 2526   TASK: 1002f82b7f0       CPU: 1   COMMAND: "multi-thread"
    PID: 2527   TASK: 1003e1737f0       CPU: 1   COMMAND: "multi-thread"
    PID: 2528   TASK: 10035b4b7f0       CPU: 1   COMMAND: "multi-thread"
    PID: 2529   TASK: 1003f0c37f0       CPU: 1   COMMAND: "multi-thread"
    PID: 2530   TASK: 10035597030       CPU: 1   COMMAND: "multi-thread"
    PID: 2531   TASK: 100184be7f0       CPU: 1   COMMAND: "multi-thread"
```

Display the resource limits of "bash" task 13896:

```
  crash> ps -r 13896
  PID: 13896  TASK: cf402000  CPU: 0   COMMAND: "bash"
     RLIMIT     CURRENT       MAXIMUM
        CPU   (unlimited)   (unlimited)
      FSIZE   (unlimited)   (unlimited)
       DATA   (unlimited)   (unlimited)
      STACK    10485760     (unlimited)
       CORE   (unlimited)   (unlimited)
        RSS   (unlimited)   (unlimited)
      NPROC      4091          4091
     NOFILE      1024          1024
    MEMLOCK      4096          4096
         AS   (unlimited)   (unlimited)
      LOCKS   (unlimited)   (unlimited)
```

Search for task names matching a POSIX regular expression:

```
   crash> ps 'migration*'
      PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
        8      2   0  ffff8802128a2e20  IN   0.0       0      0  [migration/0]
       10      2   1  ffff880212969710  IN   0.0       0      0  [migration/1]
       15      2   2  ffff880212989710  IN   0.0       0      0  [migration/2]
       20      2   3  ffff8802129a9710  IN   0.0       0      0  [migration/3]
```

NAME

struct - structure contents

SYNOPSIS

```
struct struct_name[.member[,member]][-o][-l offset][-rfuxdp]
       [address | symbol][:cpuspec] [count | -c count]
```

DESCRIPTION

This command displays either a structure definition, or a formatted display of the contents of a structure at a specified address. When no address is specified, the structure definition is shown along with the structure size. A structure member may be appended to the structure name in order to limit the scope of the data displayed to that particular member; when no address

```
  is specified, the member's offset and definition are shown.

    struct_name  name of a C-code structure used by the kernel.
        .member  name of a structure member; to display multiple members of a
                 structure, use a comma-separated list of members.  If any
                 member contains an embedded structure, or the member is an
                 array, the output may be restricted to just the embedded
                 structure or an array element by expressing the member argument
                 as "member.member" or "member[index]"; embedded member
                 specifications may extend beyond one level deep, by expressing
                 the member argument as "member.member.member...".
              -o  show member offsets when displaying structure definitions;
                  if used with an address or symbol argument, each member will
                  be preceded by its virtual address.
      -l offset  if the address argument is a pointer to a structure member that
                 is contained by the target data structure, typically a pointer
                 to an embedded list_head, the offset to the embedded member may
                 be entered in either of the following manners:
                   1. in "structure.member" format.
                   2. a number of bytes.
              -r  raw dump of structure data.
              -f  address argument is a dumpfile offset.
              -u  address argument is a user virtual address in the current
                  context.
              -x  override default output format with hexadecimal format.
              -d  override default output format with decimal format.
              -p  if a structure member is a pointer value, show the member's
                  data type on the output line; and on the subsequent line(s),
                  dereference the pointer, display the pointer target's symbol
                  value in brackets if appropriate, and if possible, display the
                  target data; requires an address argument.
         address  hexadecimal address of a structure; if the address points
                  to an embedded list_head structure contained within the
                  target data structure, then the "-l" option must be used.
          symbol  symbolic reference to the address of a structure.
       :cpuspec  CPU specification for a per-cpu address or symbol:
                   :             CPU of the currently selected task.
                   :a[ll]        all CPUs.
                   :#[-#][,...]  CPU list(s), e.g. "1,3,5", "1-3",
                                 or "1,3,5-7,10".
           count  count of structures to dump from an array of structures;
                  if used, this must be the last argument entered.
       -c count  "-c" is only required if "count" is not the last argument
                 entered or if a negative number is entered; if a negative
                 value is entered, the (positive) "count" structures that
                 lead up to and include the target structure will be displayed.

  Structure data, sizes, and member offsets are shown in the current output
  radix unless the -x or -d option is specified.

  Please note that in the vast majority of cases, the "struct" command
  name may be dropped; if the structure name does not conflict with any crash
  or gdb command name, then the "struct_name[.member]" argument will be
  recognized as a structure name, and this command automatically executed.
  See the NOTE below.

EXAMPLES
  Display the vm_area_struct at address c1e44f10:

    crash> struct vm_area_struct c1e44f10
    struct vm_area_struct {
      vm_mm = 0xc2857750,
      vm_start = 0x8048000,
      vm_end = 0x80a5000,
      vm_next = 0xc1e44a10,
      vm_page_prot = {
        pgprot = 0x25
      },
      vm_flags = 0x1875,
      vm_avl_height = 0x2,
      vm_avl_left = 0xc30fe200,
      vm_avl_right = 0xc30fed00,
      vm_next_share = 0x0,
```

```
      vm_pprev_share = 0xc1e44a30,
      vm_ops = 0xc0215ca0,
      vm_offset = 0x0,
      vm_file = 0xc0bfdc70,
      vm_pte = 0
    }
```

Display the definition and size of a vm_area_struct structure. This first example below displays just the structure and size. The second example uses the -o option to also display member offsets. Both examples were run with the output radix set to 10 (decimal):

```
  crash> struct vm_area_struct
  struct vm_area_struct {
      struct mm_struct *vm_mm;
      long unsigned int vm_start;
      long unsigned int vm_end;
      struct vm_area_struct *vm_next;
      pgprot_t vm_page_prot;
      short unsigned int vm_flags;
      short int vm_avl_height;
      struct vm_area_struct *vm_avl_left;
      struct vm_area_struct *vm_avl_right;
      struct vm_area_struct *vm_next_share;
      struct vm_area_struct **vm_pprev_share;
      struct vm_operations_struct *vm_ops;
      long unsigned int vm_offset;
      struct file *vm_file;
      long unsigned int vm_pte;
  }
  SIZE: 56

  crash> struct vm_area_struct -o
  struct vm_area_struct {
     [0] struct mm_struct *vm_mm;
     [4] long unsigned int vm_start;
     [8] long unsigned int vm_end;
    [12] struct vm_area_struct *vm_next;
    [16] pgprot_t vm_page_prot;
    [20] short unsigned int vm_flags;
    [22] short int vm_avl_height;
    [24] struct vm_area_struct *vm_avl_left;
    [28] struct vm_area_struct *vm_avl_right;
    [32] struct vm_area_struct *vm_next_share;
    [36] struct vm_area_struct **vm_pprev_share;
    [40] struct vm_operations_struct *vm_ops;
    [44] long unsigned int vm_offset;
    [48] struct file *vm_file;
    [52] long unsigned int vm_pte;
  }
  SIZE: 56
```

Display the definition and offset of the pgd member of an mm_struct:

```
  crash> struct mm_struct.pgd
  struct mm_struct {
     [80] pgd_t *pgd;
  }
```

Display the pgd member of the mm_struct at address ffff810022e7d080:

```
  crash> struct mm_struct.pgd ffff810022e7d080
    pgd = 0xffff81000e3ac000
```

Display the pgd_t pointed to by the mm_struct.pgd pointer above, forcing the output to be expressed in hexadecimal:

```
  crash> mm_struct.pgd ffff810022e7d080 -px
    pgd_t *pgd = 0xffff81000e3ac000
    -> {
         pgd = 0x2c0a6067
       }
```

```
Display the thread_info structure pointed to by the thread_info
member of the task_struct at ffff8100181190c0:

  crash> task_struct.thread_info ffff8100181190c0 -p
    struct thread_info *thread_info = 0xffff810023c06000
    -> {
         task = 0xffff8100181190c0,
         exec_domain = 0xffffffff802f78e0,
         flags = 128,
         status = 1,
         cpu = 3,
         preempt_count = 0,
         addr_limit = {
           seg = 18446604435732824064
         },
         restart_block = {
           fn = 0xffffffff80095a52 <do_no_restart_syscall>,
           arg0 = 0,
           arg1 = 0,
           arg2 = 0,
           arg3 = 0
         }
       }

Display the flags and virtual members of 4 contiguous page structures
in the mem_map page structure array:

  crash> page.flags,virtual c101196c 4
    flags = 0x8000,
    virtual = 0xc04b0000

    flags = 0x8000,
    virtual = 0xc04b1000

    flags = 0x8000,
    virtual = 0xc04b2000

    flags = 0x8000,
    virtual = 0xc04b3000

Display the array of tcp_sl_timer structures declared by tcp_slt_array[]:

  crash> struct tcp_sl_timer tcp_slt_array 4
  struct tcp_sl_timer {
    count = {
      counter = 0x0
    },
    period = 0x32,
    last = 0x1419e4,
    handler = 0xc0164854  <tcp_syn_recv_timer>
  }
  struct tcp_sl_timer {
    count = {
      counter = 0x2
    },
    period = 0x753,
    last = 0x14a6df,
    handler = 0xc01645b0  <tcp_keepalive>
  }
  struct tcp_sl_timer {
    count = {
      counter = 0x0
    },
    period = 0x2ee,
    last = 0x143134,
    handler = 0xc016447c  <tcp_twkill>
  }
  struct tcp_sl_timer {
    count = {
      counter = 0x0
    },
    period = 0x64,
    last = 0x143198,
```

```
      handler = 0xc0164404  <tcp_bucketgc>
    }

Without using the "struct" command name, display the the "d_child"
list_head member from a dentry structure:

  crash> dentry.d_child 0xe813cb4
    d_child = {
      next = 0x3661344,
      prev = 0xdea4bc4
    },

Display the child dentry structure referenced by the "next" pointer above.
Since the "next" address of 0x3661344 above is a pointer to an embedded
list_head structure within the child dentry structure, the -l option
is required:

  crash> dentry -l dentry.d_child 0x3661344
  struct dentry {
    d_count = {
      counter = 1
    },
    d_flags = 0,
    d_inode = 0xf9aa604,
    d_parent = 0x11152b1c,
    d_hash = {
      next = 0x11fb3fc0,
      prev = 0x11fb3fc0
    },
    d_lru = {
      next = 0x366133c,
      prev = 0x366133c
    },
    d_child = {
      next = 0x36613cc,
      prev = 0xe813cd4
    },
    d_subdirs = {
      next = 0x366134c,
      prev = 0x366134c
    },
    d_alias = {
      next = 0xf9aa614,
      prev = 0xf9aa614
    },
    d_mounted = 0,
    d_name = {
      name = 0x3661384 "boot.log",
      len = 8,
      hash = 1935169207
    },
    d_time = 1515870810,
    d_op = 0x0,
    d_sb = 0x11fc9c00,
    d_vfs_flags = 0,
    d_fsdata = 0x0,
    d_extra_attributes = 0x0,
    d_iname = "boot.log\000"
  }

Display the virtual address of each member of the task_struct at
ffff8100145d2080:

  crash> task_struct -o ffff8100145d2080
  struct task_struct {
    [ffff8100145d2080] volatile long int state;
    [ffff8100145d2088] struct thread_info *thread_info;
    [ffff8100145d2090] atomic_t usage;
    [ffff8100145d2098] long unsigned int flags;
    [ffff8100145d20a0] int lock_depth;
    [ffff8100145d20a4] int load_weight;
    [ffff8100145d20a8] int prio;
    [ffff8100145d20ac] int static_prio;
```

```
      [ffff8100145d20b0] int normal_prio;
      [ffff8100145d20b8] struct list_head run_list;
      [ffff8100145d20c8] struct prio_array *array;
    ...
```

Display the embedded sched_entity structure's on_rq member and the third pid_link structure in the embedded pids[] array of the task_struct at ffff88011653e250:

```
  crash> task_struct.se.on_rq,pids[2] ffff88011653e250
    se.on_rq = 1,
    pids[2] =   {
      node = {
        next = 0xffff88011653aff0,
        pprev = 0xffff88011653a860
      },
      pid = 0xffff88010d07ed00
    }
```

For an example of displaying per-cpu variables, consider the struct hd_struct.dkstats member, which is a percpu pointer to a disk_stats structure:

```
  crash> struct hd_struct.dkstats
  struct hd_struct {
    [1232] struct disk_stats *dkstats;
  }
```

Taking an hd_struct at address ffff8802450e2848, display all of the per-cpu disk_stats structures that it references:

```
  crash> struct hd_struct.dkstats ffff8802450e2848
    dkstats = 0x60fdb48026c8
  crash> struct disk_stats 0x60fdb48026c8:a
  [0]: ffffe8fefe6026c8
  struct disk_stats {
    sectors = {451376, 80468},
    ios = {6041, 971},
    merges = {386, 390},
    ticks = {194877, 56131},
    io_ticks = 12371,
    time_in_queue = 309163
  }
  [1]: ffffe8fefe8026c8
  struct disk_stats {
    sectors = {0, 0},
    ios = {0, 0},
    merges = {7, 242},
    ticks = {0, 0},
    io_ticks = 23,
    time_in_queue = 581
  }
  [2]: ffffe8fefea026c8
  struct disk_stats {
    sectors = {0, 0},
    ios = {0, 0},
    merges = {4, 112},
    ticks = {0, 0},
    io_ticks = 11,
    time_in_queue = 305
  }
  [3]: ffffe8fefec026c8
  struct disk_stats {
    sectors = {0, 0},
    ios = {0, 0},
    merges = {5, 54},
    ticks = {0, 0},
    io_ticks = 17,
    time_in_queue = 41
  }
```

NOTE

If the structure name does not conflict with any crash command name, the "struct" command may be dropped. Accordingly, the examples above could also have been accomplished like so:

```
crash> vm_area_struct c1e44f10
crash> vm_area_struct
crash> vm_area_struct -o
crash> mm_struct.pgd ffff810022e7d080
crash> mm_struct.pgd
crash> tcp_sl_timer tcp_slt_array 4
```

Lastly, the short-cut "*" pointer-to command may also be used to negate the need to enter the "struct" command name (enter "help *" for details).

NAME

waitq - list tasks queued on a wait queue

SYNOPSIS

```
waitq  [ symbol ] | [ struct.member struct_addr ] | [ address ]
```

DESCRIPTION

This command walks the wait queue list displaying the tasks which are blocked on the specified wait queue. The command differentiates between the old- and new-style wait queue structures used by the kernel. It can be invoked with the following argument types:

```
                     symbol  a global symbol of a wait queue.
struct.member struct_addr  a structure name and wait queue member combination
                           followed by the structure's hexadecimal address.
                   address  a hexadecimal wait queue pointer.
```

EXAMPLES

Find out if any tasks are blocked on the "buffer_wait" wait queue:

```
crash> waitq buffer_wait
wait queue "buffer_wait" (c02927f0) is empty
```

See who is blocked on the "wait_chldexit" queue of task c5496000:

```
crash> waitq task_struct.wait_chldexit c5496000
PID: 30879  TASK: c5496000  CPU: 0   COMMAND: "bash"
```

Display the task list waiting on a known task queue:

```
crash> waitq c3534098
PID: 13691  TASK: c3534000  CPU: 1   COMMAND: "bash"
```

NAME

btop - bytes to page

SYNOPSIS

```
btop address ...
```

DESCRIPTION

This command translates a hexadecimal address to its page number.

EXAMPLES

```
crash> btop 512a000
512a000: 512a
```

NAME

ipcs - System V IPC facilities

SYNOPSIS

```
ipcs [-smMq] [-n pid|task] [id | addr]
```

DESCRIPTION

This command provides information on the System V IPC facilities. With no arguments, the command will display kernel usage of all three facilities.

```
        -s  show semaphore arrays.
        -m  show shared memory segments.
        -M  show shared memory segments with additional details.
        -q  show message queues.
        id  show the data associated with this resource ID.
      addr  show the data associated with this virtual address of a
            shmid_kernel, sem_array or msq_queue.

  For kernels supporting namespaces, the -n option may be used to
  display the IPC facilities with respect to the namespace of a
  specified task:

  -n pid   a process PID.
  -n task  a hexadecimal task_struct pointer.

EXAMPLES
  Display all IPC facilities:

    crash> ipcs
    SHMID_KERNEL     KEY      SHMID      UID   PERMS BYTES      NATTCH STATUS
    ffff880473a28310 00000000 0          0     666   90000      1
    ffff880473a28490 00000001 32769      0     666   90000      1
    ffff880473a28250 00000002 65538      0     666   90000      1

    SEM_ARRAY        KEY      SEMID      UID   PERMS NSEMS
    ffff88047200f9d0 00000000 0          0     600   1
    ffff88046f826910 00000000 32769      0     600   1

    MSG_QUEUE        KEY      MSQID      UID   PERMS USED-BYTES   MESSAGES
    ffff8100036bb8d0 000079d7 0          3369  666   16640        104
    ffff8100036bb3d0 000079d8 32769      3369  666   12960        81
    ffff810026d751d0 000079d9 65538      3369  666   10880        68

  Display shared memory usage with detailed information:

    crash> ipcs -M
    SHMID_KERNEL     KEY      SHMID      UID   PERMS BYTES      NATTCH STATUS
    ffff880473a28310 00000000 0          0     666   90000      1
    PAGES ALLOCATED/RESIDENT/SWAPPED: 22/1/0
    INODE: ffff88047239cd98

    SHMID_KERNEL     KEY      SHMID      UID   PERMS BYTES      NATTCH STATUS
    ffff880473a28490 00000001 32769      0     666   90000      1
    PAGES ALLOCATED/RESIDENT/SWAPPED: 22/1/0
    INODE: ffff88047239c118

    SHMID_KERNEL     KEY      SHMID      UID   PERMS BYTES      NATTCH STATUS
    ffff880473a28250 00000002 65538      0     666   90000      1
    PAGES ALLOCATED/RESIDENT/SWAPPED: 22/1/0
    INODE: ffff880470503758

  Display the shared memory data associated with shmid_kernel ffff880473a28250:

    crash> ipcs -M ffff880473a28250
    SHMID_KERNEL     KEY      SHMID      UID   PERMS BYTES      NATTCH STATUS
    ffff880473a28250 00000002 65538      0     666   90000      1
    PAGES ALLOCATED/RESIDENT/SWAPPED: 22/1/0
    INODE: ffff880470503758


NAME
  pte - translate a page table entry

SYNOPSIS
  pte contents ...

DESCRIPTION
  This command translates the hexadecimal contents of a PTE into its physical
  page address and page bit settings.  If the PTE references a swap location,
  the swap device and offset are displayed.

EXAMPLES
```

```
    crash> pte d8e067
     PTE    PHYSICAL  FLAGS
    d8e067   d8e000   (PRESENT|RW|USER|ACCESSED|DIRTY)

    crash> pte 13f600
     PTE      SWAP     OFFSET
    13f600  /dev/hda2   5104


NAME
  swap - swap device information

SYNOPSIS
  swap

DESCRIPTION
  This command displays information for each configured swap device.

EXAMPLE
  crash> swap
  SWAP_INFO_STRUCT    TYPE       SIZE       USED    PCT  PRI  FILENAME
  ffff880153d45f40  PARTITION  7192568k   1200580k  16%   -1  /dev/dm-1


NAME
  whatis - search symbol table for data or type information

SYNOPSIS
  whatis [[-o] [struct | union | typedef | symbol]] |
         [[-r [size|range]] [-m member]]

DESCRIPTION
  This command displays the definition of structures, unions, typedefs or
  text/data symbols:

    struct  a structure name. The output is the same as if the "struct"
            command was used.
     union  a union name. The output is the same as if the "union" command
            was used.
        -o  display the offsets of structure/union members.
   typedef  a typedef name. If the typedef translates to a structure or union
            the output is the same as if the "struct" or "union" command
            was used. If the typedef is a primitive datatype, the one-line
            declaration is displayed.
    symbol  a kernel symbol.

  Alternatively, a search can be made for data structures of a given size or
  size range, that contain a member of a given type, or contain a pointer to
  given type.  The -r and -m options may be used alone or in conjunction with
  one another:

    -r size  search for structures of this exact size.
   -r range  search for structures of a range of sizes, expressed as "low-high".
  -m member  search for structures that contain a member of this data type, or
            that contain a pointer to this data type; if a structure contains
            another structure, the members of the embedded structure will also
            be subject to the search. The member argument may also be expressed
            as a substring of a member's data type.

EXAMPLES
   Display the definition of a linux_binfmt structure:

    crash> whatis linux_binfmt
    struct linux_binfmt {
        struct list_head lh;
        struct module *module;
        int (*load_binary)(struct linux_binprm *);
        int (*load_shlib)(struct file *);
        int (*core_dump)(struct coredump_params *);
        unsigned long min_coredump;
    }
    SIZE: 56
```

```
 Display the same structure with member offsets:

  crash> whatis -o linux_binfmt
  struct linux_binfmt {
     [0] struct list_head lh;
    [16] struct module *module;
    [24] int (*load_binary)(struct linux_binprm *);
    [32] int (*load_shlib)(struct file *);
    [40] int (*core_dump)(struct coredump_params *);
    [48] unsigned long min_coredump;
  }
  SIZE: 56

Since a kmem_bufctl_t is typedef'd to be a kmem_bufctl_s structure, the
output of the following two commands is identical:

  crash> whatis kmem_bufctl_s
  struct kmem_bufctl_s {
    union {
      struct kmem_bufctl_s  *buf_nextp;
      kmem_slab_t *buf_slabp;
      void *buf_objp;
    } u;
  };

  crash> whatis kmem_bufctl_t
  struct kmem_bufctl_s {
    union {
      struct kmem_bufctl_s *buf_nextp;
      kmem_slab_t *buf_slabp;
      void *buf_objp;
    } u;
  };
  SIZE: 4  (0x4)

Display the type data of sys_read() and jiffies text and data symbols:

  crash> whatis sys_read
  ssize_t sys_read(unsigned int, char *, size_t);

  crash> whatis jiffies
  long unsigned int jiffies;

Display definition of a kdev_t typedef:

  crash> whatis kdev_t
  typedef short unsigned int kdev_t;
  SIZE: 2  (0x2)

Display all structures which have a size of 192 bytes:

  crash> whatis -r 192
  SIZE  TYPE
   192  _intel_private
   192  blkcg_gq
   192  clock_event_device
   192  cper_sec_proc_generic
   192  dentry
   192  dst_ops
   192  ehci_itd
   192  ethtool_rxnfc
   192  fb_ops
   192  file_lock
   192  inode_operations
   192  input_device_id
   192  ip_vs_stats
   192  numa_group
   192  parallel_data
   192  pcie_port_service_driver
   192  pebs_record_hsw
   192  pnp_driver
   192  regmap_config
```

```
       192  sched_entity
       192  tcp_timewait_sock
       192  timerfd_ctx
       192  tpm_vendor_specific
       192  urb

  Display all structures that contain members that point to
  an mm_struct:

    crash> whatis -m mm_struct
    SIZE  TYPE
      16  tlb_state
      24  flush_tlb_info
      24  ftrace_raw_xen_mmu_pgd
      24  futex_key
      24  map_info
      32  ftrace_raw_xen_mmu_alloc_ptpage
      32  ftrace_raw_xen_mmu_pte_clear
      40  ftrace_raw_xen_mmu_flush_tlb_others
      40  ftrace_raw_xen_mmu_ptep_modify_prot
      40  ftrace_raw_xen_mmu_set_pte_at
      40  mm_slot
      64  mm_walk
      64  rmap_item
     104  userfaultfd_ctx
     128  mmu_gather
     216  vm_area_struct
     256  linux_binprm
    2616  rq
    2936  task_struct

  Display all structures sized from 256 to 512 bytes that
  contain members that point to a task_struct:

    crash> whatis -r 256-512 -m task_struct
    SIZE  TYPE
     256  file
     256  od_cpu_dbs_info_s
     264  srcu_notifier_head
     272  protection_domain
     288  clk_notifier
     288  fsnotify_group
     296  quota_info
     312  tty_port
     320  workqueue_struct
     344  trace_array
     344  uart_state
     352  cpufreq_policy
     352  elf_thread_core_info
     376  perf_event_context
     384  rcu_data
     400  cgroup
     408  subsys_private
     424  hvc_struct
     496  psmouse


NAME
  dev - device data

SYNOPSIS
  dev [-i | -p | -d | -D ] [-V | -v index [file]]

DESCRIPTION
  If no argument is entered, this command dumps character and block
  device data.

    -i  display I/O port usage; on 2.4 kernels, also display I/O memory usage.
    -p  display PCI device data.
    -d  display disk I/O statistics:
         TOTAL: total number of allocated in-progress I/O requests
          SYNC: I/O requests that are synchronous
         ASYNC: I/O requests that are asynchronous
```

```
            READ: I/O requests that are reads (older kernels)
           WRITE: I/O requests that are writes (older kernels)
             DRV: I/O requests that are in-flight in the device driver.
                  If the device driver uses blk-mq interface, this field
                  shows N/A(MQ).  If not available, this column is not shown.
      -D  same as -d, but filter out disks with no in-progress I/O requests.

    If the dumpfile contains device dumps:
          -V  display an indexed list of all device dumps present in the vmcore,
              showing their file offset, size and name.
    -v index  select and display one device dump based upon an index value
              shown by the -V option, shown in a default human-readable format;
              alternatively, the "rd -f" option along with its various format
              options may be used to further tailor the output.
        file  only used with -v, copy the device dump data to a file.

EXAMPLES
  Display character and block device data:

    crash> dev
    CHRDEV    NAME              CDEV    OPERATIONS
       1      mem              f79b83c0  memory_fops
       4      /dev/vc/0        c07bc560  console_fops
       4      tty              f7af5004  tty_fops
       4      ttyS             f7b02204  tty_fops
       5      /dev/tty         c07bc440  tty_fops
       5      /dev/console     c07bc4a0  console_fops
       5      /dev/ptmx        c07bc500  ptmx_fops
       6      lp               c5797e40  lp_fops
       7      vcs              f7b03d40  vcs_fops
      10      misc             f7f68640  misc_fops
      13      input            f79b8840  input_fops
      21      sg               f7f12840  sg_fops
      29      fb               f7f8c640  fb_fops
     128      ptm              f7b02604  tty_fops
     136      pts              f7b02404  tty_fops
     162      raw              c0693e40  raw_fops
     180      usb              f79b8bc0  usb_fops
     189      usb_device       c06a0300  usbfs_device_file_operations
     216      rfcomm           f5961a04  tty_fops
     254      pcmcia           f79b82c0  ds_fops

    BLKDEV    NAME              GENDISK  OPERATIONS
       1      ramdisk          f7b23480  rd_bd_op
       8      sd               f7cab280  sd_fops
       9      md               f7829b80  md_fops
      11      sr               f75c24c0  sr_bdops
      65      sd                (none)
      66      sd                (none)
      67      sd                (none)
      68      sd                (none)
      69      sd                (none)
      70      sd                (none)
      71      sd                (none)
     128      sd                (none)
     129      sd                (none)
     130      sd                (none)
     131      sd                (none)
     132      sd                (none)
     133      sd                (none)
     134      sd                (none)
     135      sd                (none)
     253      device-mapper    c57a0ac0  dm_blk_dops
     254      mdp               (none)

  Display PCI data:

    crash> dev -p
    PCI_DEV  BU:SL.FN CLASS: VENDOR-DEVICE
    c00051c0 00:00.0  Host bridge: Intel 440BX - 82443BX Host
    c0005250 00:01.0  PCI bridge: Intel 440BX - 82443BX AGP
    c00052e0 00:07.0  ISA bridge: Intel 82371AB PIIX4 ISA
    c0005370 00:07.1  IDE interface: Intel 82371AB PIIX4 IDE
```

```
  c0005400 00:07.2  USB Controller: Intel 82371AB PIIX4 USB
  c0005490 00:07.3  Bridge: Intel 82371AB PIIX4 ACPI
  c0005520 00:11.0  Ethernet controller: 3Com 3C905B 100bTX
  c00055b0 00:13.0  PCI bridge: DEC DC21152
  c0005640 01:00.0  VGA compatible controller: NVidia [PCI_DEVICE 28]
  c00056d0 02:0a.0  SCSI storage controller: Adaptec AIC-7890/1
  c0005760 02:0e.0  SCSI storage controller: Adaptec AIC-7880U
```

Display I/O port and I/O memory usage:

```
  crash> dev -i
  RESOURCE    RANGE    NAME
  c03036d4  0000-ffff  PCI IO
  c0302594  0000-001f  dma1
  c03025b0  0020-003f  pic1
  c03025cc  0040-005f  timer
  c03025e8  0060-006f  keyboard
  c0302604  0080-008f  dma page reg
  c0302620  00a0-00bf  pic2
  c030263c  00c0-00df  dma2
  c0302658  00f0-00ff  fpu
  c122ff20  0170-0177  ide1
  c122f240  0213-0213  isapnp read
  c122ff40  02f8-02ff  serial(auto)
  c122ff00  0376-0376  ide1
  c03186e8  03c0-03df  vga+
  c122ff60  03f8-03ff  serial(auto)
  c123851c  0800-083f  Intel Corporation 82371AB PIIX4 ACPI
  c1238538  0840-085f  Intel Corporation 82371AB PIIX4 ACPI
  c122f220  0a79-0a79  isapnp write
  c122f200  0cf8-0cff  PCI conf1
  c1238858  dc00-dc7f  3Com Corporation 3c905B 100BaseTX [Cyclone]
  c122fc00  dc00-dc7f  00:11.0
  c12380c8  dce0-dcff  Intel Corporation 82371AB PIIX4 USB
  c1238d1c  e000-efff  PCI Bus #02
  c1237858  e800-e8ff  Adaptec AIC-7880U
  c1237458  ec00-ecff  Adaptec AHA-2940U2/W / 7890
  c1239cc8  ffa0-ffaf  Intel Corporation 82371AB PIIX4 IDE

  RESOURCE        RANGE        NAME
  c03036f0  00000000-ffffffff  PCI mem
  c0004000  00000000-0009ffff  System RAM
  c03026ac  000a0000-000bffff  Video RAM area
  c03026fc  000c0000-000c7fff  Video ROM
  c0302718  000c9800-000cdfff  Extension ROM
  c0302734  000ce000-000ce7ff  Extension ROM
  c0302750  000ce800-000cffff  Extension ROM
  c03026e0  000f0000-000fffff  System ROM
  c0004040  00100000-07ffdfff  System RAM
  c0302674  00100000-0028682b  Kernel code
  c0302690  0028682c-0031c63f  Kernel data
  c0004060  07ffe000-07ffffff  reserved
  c1239058  ec000000-efffffff  Intel Corporation 440BX/ZX - 82443BX/ZX Host
                               bridge
  c1238d54  f1000000-f1ffffff  PCI Bus #02
  c1239554  f2000000-f5ffffff  PCI Bus #01
  c1237074  f4000000-f5ffffff  nVidia Corporation Riva TnT2 [NV5]
  c1238d38  fa000000-fbffffff  PCI Bus #02
  c1237874  faffe000-faffefff  Adaptec AIC-7880U
  c127ec40  faffe000-faffefff  aic7xxx
  c1237474  fafff000-fafffffe  Adaptec AHA-2940U2/W / 7890
  c127eec0  fafff000-fafffffe  aic7xxx
  c1239538  fc000000-fdffffff  PCI Bus #01
  c1237058  fc000000-fcffffff  nVidia Corporation Riva TnT2 [NV5]
  c1238874  fe000000-fe00007f  3Com Corporation 3c905B 100BaseTX [Cyclone]
  c0004080  fec00000-fec0ffff  reserved
  c00040a0  fee00000-fee0ffff  reserved
  c00040c0  ffe00000-ffffffff  reserved
```

Display disk I/O statistics:

```
  crash> dev -d
  MAJOR GENDISK            NAME     REQUEST_QUEUE      TOTAL  READ WRITE   DRV
```

```
          2 ffff81012d8a5000   fd0      ffff81012dc053c0      12     0    12     0
         22 ffff81012dc6b000   hdc      ffff81012d8ae340       2     2     0     0
          8 ffff81012dd71000   sda      ffff81012d8af040       6     0     6     6
          8 ffff81012dc77000   sdb      ffff81012d8b5740       0     0     0     0
          8 ffff81012d8d0c00   sdc      ffff81012d8ae9c0       0     0     0     0
```

Display the available device dumps:

```
  crash> dev -V
  INDEX  OFFSET             SIZE             NAME
    0    0x240              33558464         cxgb4_0000:02:00.4
    1    0x2001240          33558464         cxgb4_0000:03:00.4
```

Extract a specified device dump to file:

```
  crash> dev -v 0 device_dump_0.bin
  DEVICE: cxgb4_0000:02:00.4
  33558464 bytes copied from 0x240 to device_dump_0.bin
```

Format and display a device's dump data to the screen using the "rd" command:

```
  crash> rd -f 0x240 -32 8
  240:  040b69e2 00000038 000e0001 00675fd4   .i..8........_g.
  250:  00000000 21600047 00000000 00000000   ....G.`!........
```

Display a device's dump data to the screen using the default format:

```
  crash> dev -v 1
  DEVICE: cxgb4_0000:03:00.4
           2001240:  00000038040b69e2 00af985c000e0001   .i..8.......\...
           2001250:  2150004700000000 0000000000000000   ....G.P!........
           2001260:  0000000000000000 0000000000000000   ................
           2001270:  0000000000000000 0002fccc00000001   ................
           2001280:  00000000000027b0 0000000000000000   .'..............
  ...
```

NAME

irq - IRQ data

SYNOPSIS

```
irq [[[index ...] | -u ] | -d | -b | -a | -s [-c cpu]]
```

DESCRIPTION

This command collaborates the data in an irq_desc_t, along with its associated hw_interrupt_type and irqaction structure data, into a consolidated per-IRQ display. For kernel versions 2.6.37 and later the display consists of the irq_desc/irq_data address, its irqaction address(es), and the irqaction name strings. Alternatively, the intel interrupt descriptor table, bottom half data, cpu affinity for in-use irqs, or kernel irq stats may be displayed. If no index value argument(s) nor any options are entered, the IRQ data for all IRQs will be displayed.

```
  index   a valid IRQ index.
     -u   dump data for in-use IRQs only.
     -d   dump the intel interrupt descriptor table.
     -b   dump bottom half data.
     -a   dump cpu affinity for in-use IRQs.
     -s   dump the kernel irq stats; if no cpu specified with -c, the
          irq stats of all cpus will be displayed.
 -c cpu   only usable with the -s option, dump the irq stats of the
          specified cpu[s]; cpu can be specified as "1,3,5", "1-3",
          "1,3,5-7,10", "all", or "a" (shortcut for "all").
```

EXAMPLES

Display the relevant data for IRQ 18 from a pre-2.6.37 kernel:

```
  crash> irq 18
      IRQ: 18
   STATUS: 0
  HANDLER: c02301e0  <ioapic_level_irq_type>
           typename: c01f9e0c  "IO-APIC-level"
```

```
               startup: c0110234  <unmask_IO_APIC_irq>
              shutdown: c01101cc  <mask_IO_APIC_irq>
                handle: c0110518  <do_level_ioapic_IRQ>
                enable: c0110234  <unmask_IO_APIC_irq>
               disable: c01101cc  <mask_IO_APIC_irq>
     ACTION: c009c6b0
               handler: c01ce818  <do_aic7xxx_isr>
                 flags: 4000000  (SA_SHIRQ)
                  mask: 0
                  name: c0217780  "aic7xxx"
                dev_id: c0090078
                  next: c009c770
     ACTION: c009c770
               handler: c01ce818  <do_aic7xxx_isr>
                 flags: 4000000  (SA_SHIRQ)
                  mask: 0
                  name: c0217780  "aic7xxx"
                dev_id: c0091078
                  next: 0
      DEPTH: 0
```

Display the relevant data for IRQ 21 from a 2.6.37 kernel:

```
  crash> irq 21
    IRQ   IRQ_DESC/_DATA      IRQACTION      NAME
    21   ffff88003787f780  ffff8800379a8b40  "ehci_hcd:usb2"
                           ffff8800379cbac0  "uhci_hcd:usb5"
                           ffff8800379cb140  "uhci_hcd:usb7"
```

Display the intel interrupt descriptor table entries:

```
  crash> irq -d
    [0] divide_error
    [1] debug
    [2] nmi
    [3] int3
    [4] overflow
    [5] bounds
    [6] invalid_op
    [7] device_not_available
    [8] double_fault
    [9] coprocessor_segment_overrun
   [10] invalid_TSS
   [11] segment_not_present
   [12] stack_segment
   [13] general_protection
   [14] page_fault
   [15] spurious_interrupt_bug
   [16] coprocessor_error
   [17] alignment_check
   [18] ignore_int
   [19] ignore_int
   [20] ignore_int
   [21] ignore_int
  ...

  [250] IRQ0xda_interrupt
  [251] IRQ0xdb_interrupt
  [252] IRQ0xdc_interrupt
  [253] IRQ0xdd_interrupt
  [254] IRQ0xde_interrupt
  [255] spurious_interrupt
```

Display the bottom half data:

```
  crash> irq -b
  SOFTIRQ_VEC      ACTION
      [0]     ffffffff81068f60  <tasklet_hi_action>
      [1]     ffffffff81071b80  <run_timer_softirq>
      [2]     ffffffff813e6f30  <net_tx_action>
      [3]     ffffffff813ee370  <net_rx_action>
      [4]     ffffffff81211a60  <blk_done_softirq>
      [5]     ffffffff812122f0  <blk_iopoll_softirq>
```

```
          [6]     ffffffff81069090  <tasklet_action>
          [7]     ffffffff81058830  <run_rebalance_domains>
          [8]     ffffffff81087f00  <run_hrtimer_softirq>
          [9]     ffffffff810ca7a0  <rcu_process_callbacks>
```

Display the cpu affinity for in-use IRQs:

```
  crash> irq -a
  IRQ NAME                 AFFINITY
    0 timer                0-23
    1 i8042                0-23
    8 rtc0                 0-23
    9 acpi                 0-23
   16 ehci_hcd:usb2,uhci_hcd:usb3,uhci_hcd:usb6 0,6,18
   17 uhci_hcd:usb4,uhci_hcd:usb7 0-23
   18 ehci_hcd:usb1,uhci_hcd:usb5,uhci_hcd:usb8,ioc0 0,11,23
   24 dmar0                0
   35 pciehp               0-23
   36 pciehp               0-23
   37 pciehp               0-23
   38 pciehp               0-23
   39 megasas              0-5,12-17
   40 lpfc:sp              0-5,12-17
   41 lpfc:fp              0,6-11,18-23
   42 lpfc:sp              0,6-11,18-23
   43 lpfc:fp              0,6-11,18-23
  ...

   80 ioat-msix            0-23
   81 ioat-msix            0-23
   82 ioat-msix            0-23
   83 ioat-msix            0-23
   84 ioat-msix            0-23
   85 ioat-msix            0-23
   86 ioat-msix            0-23
   87 ioat-msix            0-23
   88 eth4                 0,17
```

Display the kernel irq stats:

```
  crash>irq -c 0,2 -s
              CPU0       CPU2
    0: 2068161471          0 IR-IO-APIC-edge     timer
    1:          9          0 IR-IO-APIC-edge     i8042
    8:          1          0 IR-IO-APIC-edge     rtc0
    9:          0          0 IR-IO-APIC-fasteoi  acpi
   16:         36          0 IR-IO-APIC-fasteoi  ehci_hcd:usb2
  ...

   85:          3          0 IR-PCI-MSI-edge     ioat-msix
   86:          3          0 IR-PCI-MSI-edge     ioat-msix
   87:          3          0 IR-PCI-MSI-edge     ioat-msix
   88:         24        295 IR-PCI-MSI-edge     eth4
```

NAME
  ptob - page to bytes

SYNOPSIS
  ptob page_number ...

DESCRIPTION
  This command translates a page frame number to its byte value.

EXAMPLES
```
    crash> ptob 512a
    512a: 512a000
```

NAME
  sym - translate a symbol to its virtual address, or vice-versa

SYNOPSIS

```
  sym [-l] | [-M] | [-m module] | [-p|-n] | [-q string] | [symbol | vaddr]

DESCRIPTION
  This command translates a symbol to its virtual address, or a static
  kernel virtual address to its symbol -- or to a symbol-plus-offset value,
  if appropriate.  Additionally, the symbol type is shown in parentheses,
  and if the symbol is a known text value, the file and line number are shown.

              -l  dumps all symbols and their values.
              -M  dumps the current set of module symbols.
       -m module  dumps the current set of symbols for a specified module.
              -p  display the target symbol and the previous symbol.
              -n  display the target symbol and the next symbol.
       -q string  searches for all symbols containing "string".
          symbol  a kernel text or data symbol.
           vaddr  a kernel virtual address.

  If the "symbol", "vaddr" or "string" argument resolves to a module
  symbol, then the module name will be displayed in brackets following the
  symbol value.

EXAMPLES
  Translate data symbol jiffies to its value, and vice-versa:

    crash> sym jiffies
    c0213ae0 (D) jiffies

    crash> sym c0213ae0
    c0213ae0 (D) jiffies

  Translate a text address to its symbolic value and source file:

    crash> sym c0109944
    c0109944 (T) system_call+0x34  ../linux-2.2.5/arch/i386/kernel/signal.c: 723

  Dump the whole symbol table:

    crash> sym -l
    c0100000 (T) _stext
    c0100000 (A) _text
    c0100000 (t) startup_32
    c0100000 (T) stext
    c01000a4 (t) checkCPUtype
    c0100139 (t) is486
    c0100148 (t) is386
    c01001b1 (t) L6
    c01001b3 (t) ready
    c01001b4 (t) check_x87
    c01001da (t) setup_idt
    c01001f7 (t) rp_sidt
    c0100204 (T) stack_start
    c010020c (t) int_msg
    c0100220 (t) ignore_int
    c0100242 (t) idt_descr
    c0100244 (T) idt
    c010024a (t) gdt_descr
    c010024c (T) gdt
    c0101000 (T) swapper_pg_dir
    c0102000 (T) pg0
    c0103000 (T) empty_bad_page
    c0104000 (T) empty_bad_page_table
    c0105000 (T) empty_zero_page
    ...

  Find all symbols containing the string "pipe":

    crash> sym -q pipe
    c010ec60 (T) sys_pipe
    c012f660 (t) pipe_read
    c012f7b8 (t) pipe_write
    c012f9c0 (t) pipe_lseek
    c012f9d0 (t) bad_pipe_r
    c012f9dc (t) bad_pipe_w
```

```
  c012f9e8 (t) pipe_ioctl
  c012fa18 (t) pipe_poll
  c012fb00 (t) pipe_release
  c012fb48 (t) pipe_read_release
  c012fb5c (t) pipe_write_release
  c012fb70 (t) pipe_rdwr_release
  c012fba0 (t) pipe_read_open
  c012fbb0 (t) pipe_write_open
  c012fbc0 (t) pipe_rdwr_open
  c012fbec (t) get_pipe_inode
  c012fcc4 (T) do_pipe
  c023a920 (D) read_pipe_fops
  c023a960 (D) write_pipe_fops
  c023a9a0 (D) rdwr_pipe_fops
  c023a9e0 (D) pipe_inode_operations
```

Dump the symbols of the uart401 module, both before, and then after, the complete set of symbols are loaded with the "mod -s" command:

```
  crash> sym -m uart401
  c8032000 MODULE START: uart401
  c8032138 (?) uart401intr
  c803235c (?) attach_uart401
  c8032638 (?) probe_uart401
  c80326d4 (?) unload_uart401
  c8033770 MODULE END: uart401
  crash> mod -s uart401
   MODULE   NAME         SIZE  OBJECT FILE
  c8032000  uart401      6000  /lib/modules/2.2.14/misc/uart401.o
  crash> sym -m uart401
  c8032000 MODULE START: uart401
  c8032050 (t) my_notifier_call
  c8032084 (t) uart401_status
  c8032098 (t) uart401_cmd
  c80320a8 (t) uart401_read
  c80320bc (t) uart401_write
  c80320cc (t) uart401_input_loop
  c8032138 (T) uart401intr
  c8032168 (t) uart401_open
  c80321c8 (t) uart401_close
  c80321f4 (t) uart401_out
  c80322ac (t) uart401_start_read
  c80322b4 (t) uart401_end_read
  c80322bc (t) uart401_kick
  c80322c4 (t) uart401_buffer_status
  c80322cc (t) enter_uart_mode
  c803235c (T) attach_uart401
  c803259c (t) reset_uart401
  c8032638 (T) probe_uart401
  c80326d4 (T) unload_uart401
  c8032760 (T) init_module
  c80327cc (T) cleanup_module
  c8032b00 (d) sound_notifier
  c8032b0c (d) detected_devc
  c8032b20 (d) std_synth_info
  c8032bc0 (d) std_midi_synth
  c8033600 (d) uart401_operations
  c80336c4 (D) io
  c80336c8 (D) irq
  c80336e0 (b) hw_info.508
  c8033770 MODULE END: uart401
```

Display the value of jiffies, along with the next and previous symbols:

```
  crash> sym -np jiffies
  c023027c (D) prof_shift
  c0230280 (D) jiffies
  c02302a0 (D) task
```

Translate a symbol value to its name and module:

```
  crash> sym f88878d1
  f88878d1 (t) ext3_readdir [ext3]
```

```
    crash>


NAME
  wr - write memory

SYNOPSIS
  wr [-u|-k|-p] [-8|-16|-32|-64] [address|symbol] value

DESCRIPTION
  This command modifies the contents of memory.  The starting address may be
  entered either symbolically or by address.  The default modification size
  is the size of a long data type.  Write permission must exist on the
  /dev/mem.  When writing to memory on a live system, this command should
  obviously be used with great care.

       -u  address argument is a user virtual address.
       -k  address argument is a kernel virtual address.
       -p  address argument is a physical address.
       -8  write data in an 8-bit value.
      -16  write data in a 16-bit value.
      -32  write data in a 32-bit values (default on 32-bit machines).
      -64  write data in a 64-bit values (default on 64-bit machines).
  address  address to write.  The address is considered virtual unless the
           -p option is used.  If a virtual address is specified, the
           -u or -k options are necessary only if the address space cannot
           be determined from the address value itself.  If a user virtual
           address is specified, the address space of the current context
           implied.  The address must be expressed in hexadecimal format.
   symbol  symbol of starting address to write.
    value  the value of the data to write.

EXAMPLES
  Turn on a debug flag:

    crash> wr my_debug_flag 1


NAME
  dis - disassemble

SYNOPSIS
  dis [-rfludxs][-b [num]] [address | symbol | (expression)] [count]

DESCRIPTION
  This command disassembles source code instructions starting (or ending) at
  a text address that may be expressed by value, symbol or expression:

            -r  (reverse) displays all instructions from the start of the
                routine up to and including the designated address.
            -f  (forward) displays all instructions from the given address
                to the end of the routine.
            -l  displays source code line number data in addition to the
                disassembly output.
            -u  address is a user virtual address in the current context;
                otherwise the address is assumed to be a kernel virtual address.
                If this option is used, then -r and -l are ignored.
            -x  override default output format with hexadecimal format.
            -d  override default output format with decimal format.
            -s  displays the filename and line number of the source code that
                is associated with the specified text location, followed by a
                source code listing if it is available on the host machine.
                The line associated with the text location will be marked with
                an asterisk; depending upon gdb's internal "listsize" variable,
                several lines will precede the marked location. If a "count"
                argument is entered, it specifies the number of source code
                lines to be displayed after the marked location; otherwise
                the remaining source code of the containing function will be
                displayed.
       -b [num]  modify the pre-calculated number of encoded bytes to skip after
                a kernel BUG ("ud2a") instruction; with no argument, displays
                the current number of bytes being skipped. (x86 and x86_64 only)
        address  starting hexadecimal text address.
```

```
          symbol  symbol of starting text address.  On ppc64, the symbol
                  preceded by '.' is used.
    (expression)  expression evaluating to a starting text address.
           count  the number of instructions to be disassembled (default is 1).
                  If no count argument is entered, and the starting address
                  is entered as a text symbol, then the whole routine will be
                  disassembled.  The count argument is supported when used with
                  the -r and -f options.

EXAMPLES
  Disassemble the sys_signal() routine without, and then with, line numbers:

    crash> dis sys_signal
    0xc0112c88 <sys_signal>:        push   %ebp
    0xc0112c89 <sys_signal+1>:      mov    %esp,%ebp
    0xc0112c8b <sys_signal+3>:      sub    $0x28,%esp
    0xc0112c8e <sys_signal+6>:      mov    0xc(%ebp),%eax
    0xc0112c91 <sys_signal+9>:      mov    %eax,0xfffffffec(%ebp)
    0xc0112c94 <sys_signal+12>:     movl   $0xc0000000,0xfffffff0(%ebp)
    0xc0112c9b <sys_signal+19>:     lea    0xfffffd8(%ebp),%eax
    0xc0112c9e <sys_signal+22>:     push   %eax
    0xc0112c9f <sys_signal+23>:     lea    0xfffffec(%ebp),%eax
    0xc0112ca2 <sys_signal+26>:     push   %eax
    0xc0112ca3 <sys_signal+27>:     pushl  0x8(%ebp)
    0xc0112ca6 <sys_signal+30>:     call   0xc01124b8 <do_sigaction>
    0xc0112cab <sys_signal+35>:     test   %eax,%eax
    0xc0112cad <sys_signal+37>:     jne    0xc0112cb2 <sys_signal+42>
    0xc0112caf <sys_signal+39>:     mov    0xfffffd8(%ebp),%eax
    0xc0112cb2 <sys_signal+42>:     leave
    0xc0112cb3 <sys_signal+43>:     ret

    crash> dis -l sys_signal
    /usr/src/linux-2.2.5/kernel/signal.c: 1074
    0xc0112c88 <sys_signal>:        push   %ebp
    0xc0112c89 <sys_signal+1>:      mov    %esp,%ebp
    0xc0112c8b <sys_signal+3>:      sub    $0x28,%esp
    0xc0112c8e <sys_signal+6>:      mov    0xc(%ebp),%eax
    /usr/src/linux-2.2.5/kernel/signal.c: 1078
    0xc0112c91 <sys_signal+9>:      mov    %eax,0xfffffffec(%ebp)
    /usr/src/linux-2.2.5/kernel/signal.c: 1079
    0xc0112c94 <sys_signal+12>:     movl   $0xc0000000,0xfffffff0(%ebp)
    /usr/src/linux-2.2.5/kernel/signal.c: 1081
    0xc0112c9b <sys_signal+19>:     lea    0xfffffd8(%ebp),%eax
    0xc0112c9e <sys_signal+22>:     push   %eax
    0xc0112c9f <sys_signal+23>:     lea    0xfffffec(%ebp),%eax
    0xc0112ca2 <sys_signal+26>:     push   %eax
    0xc0112ca3 <sys_signal+27>:     pushl  0x8(%ebp)
    0xc0112ca6 <sys_signal+30>:     call   0xc01124b8 <do_sigaction>
    /usr/src/linux-2.2.5/kernel/signal.c: 1083
    0xc0112cab <sys_signal+35>:     test   %eax,%eax
    0xc0112cad <sys_signal+37>:     jne    0xc0112cb2 <sys_signal+42>
    0xc0112caf <sys_signal+39>:     mov    0xfffffd8(%ebp),%eax
    /usr/src/linux-2.2.5/kernel/signal.c: 1084
    0xc0112cb2 <sys_signal+42>:     leave
    0xc0112cb3 <sys_signal+43>:     ret

  Given a return address expression of "do_no_page+65", find out the
  function that do_no_page() calls by using the reverse flag:

    crash> dis -r (do_no_page+65)
    0xc011ea68 <do_no_page>:        push   %ebp
    0xc011ea69 <do_no_page+1>:      mov    %esp,%ebp
    0xc011ea6b <do_no_page+3>:      push   %edi
    0xc011ea6c <do_no_page+4>:      push   %esi
    0xc011ea6d <do_no_page+5>:      push   %ebx
    0xc011ea6e <do_no_page+6>:      mov    0xc(%ebp),%ebx
    0xc011ea71 <do_no_page+9>:      mov    0x10(%ebp),%edx
    0xc011ea74 <do_no_page+12>:     mov    0x14(%ebp),%edi
    0xc011ea77 <do_no_page+15>:     mov    0x28(%ebx),%eax
    0xc011ea7a <do_no_page+18>:     test   %eax,%eax
    0xc011ea7c <do_no_page+20>:     je     0xc011ea85 <do_no_page+29>
    0xc011ea7e <do_no_page+22>:     mov    0x18(%eax),%ecx
    0xc011ea81 <do_no_page+25>:     test   %ecx,%ecx
```

```
  0xc011ea83 <do_no_page+27>:     jne    0xc011eab0 <do_no_page+72>
  0xc011ea85 <do_no_page+29>:     mov    $0xffffe000,%eax
  0xc011ea8a <do_no_page+34>:     and    %esp,%eax
  0xc011ea8c <do_no_page+36>:     decl   0x30(%eax)
  0xc011ea8f <do_no_page+39>:     jns    0xc011ea9a <do_no_page+50>
  0xc011ea91 <do_no_page+41>:     lock btrl $0x0,0xc022fb60
  0xc011ea9a <do_no_page+50>:     push   %edi
  0xc011ea9b <do_no_page+51>:     mov    0x18(%ebp),%esi
  0xc011ea9e <do_no_page+54>:     push   %esi
  0xc011ea9f <do_no_page+55>:     push   %ebx
  0xc011eaa0 <do_no_page+56>:     mov    0x8(%ebp),%esi
  0xc011eaa3 <do_no_page+59>:     push   %esi
  0xc011eaa4 <do_no_page+60>:     call   0xc011e9e4 <do_anonymous_page>
  0xc011eaa9 <do_no_page+65>:     jmp    0xc011eb47 <do_no_page+223>
```

Disassemble 10 instructions starting at user virtual address 0x81ec624:

```
  crash> dis -u 81ec624 10
  0x81ec624:      push   %ebp
  0x81ec625:      mov    %esp,%ebp
  0x81ec627:      sub    $0x18,%esp
  0x81ec62a:      movl   $0x1,0x8(%ebp)
  0x81ec631:      mov    0x82f9040,%eax
  0x81ec636:      mov    0x10(%eax),%edx
  0x81ec639:      and    $0x100,%edx
  0x81ec63f:      mov    0x14(%eax),%ecx
  0x81ec642:      and    $0x0,%ecx
  0x81ec645:      mov    %ecx,%eax
```

Override the current decimal output radix format:

```
  crash> dis sys_read 10 -x
  0xffffffff8001178f <sys_read>:  push   %r13
  0xffffffff80011791 <sys_read+0x2>:    mov    %rsi,%r13
  0xffffffff80011794 <sys_read+0x5>:    push   %r12
  0xffffffff80011796 <sys_read+0x7>:    mov    $0xfffffffffffffff7,%r12
  0xffffffff8001179d <sys_read+0xe>:    push   %rbp
  0xffffffff8001179e <sys_read+0xf>:    mov    %rdx,%rbp
  0xffffffff800117a1 <sys_read+0x12>:   push   %rbx
  0xffffffff800117a2 <sys_read+0x13>:   sub    $0x18,%rsp
  0xffffffff800117a6 <sys_read+0x17>:   lea    0x14(%rsp),%rsi
  0xffffffff800117ab <sys_read+0x1c>:   callq  0xffffffff8000b5b4 <fget_light>
```

Disassemble from vfs_read+320 until the end of the function:

```
  crash> dis -f vfs_read+320
  0xffffffff8119d4e0 <vfs_read+320>:  cmpq   $0x0,0x20(%rax)
  0xffffffff8119d4e5 <vfs_read+325>:  jne    0xffffffff8119d3e8 <vfs_read+72>
  0xffffffff8119d4eb <vfs_read+331>:  mov    $0xffffffffffffffea,%r12
  0xffffffff8119d4f2 <vfs_read+338>:  jmp    0xffffffff8119d4c3 <vfs_read+291>
  0xffffffff8119d4f4 <vfs_read+340>:  nopl   0x0(%rax)
  0xffffffff8119d4f8 <vfs_read+344>:  callq  0xffffffff8119cc40 <do_sync_read>
  0xffffffff8119d4fd <vfs_read+349>:  mov    %rax,%r12
  0xffffffff8119d500 <vfs_read+352>:  jmpq   0xffffffff8119d44c <vfs_read+172>
  0xffffffff8119d505 <vfs_read+357>:  nopl   (%rax)
  0xffffffff8119d508 <vfs_read+360>:  mov    $0xfffffffffffffff7,%r12
  0xffffffff8119d50f <vfs_read+367>:  jmp    0xffffffff8119d4c3 <vfs_read+291>
  0xffffffff8119d511 <vfs_read+369>:  mov    $0xfffffffffffffff2,%r12
  0xffffffff8119d518 <vfs_read+376>:  jmp    0xffffffff8119d4c3 <vfs_read+291>
  0xffffffff8119d51a <vfs_read+378>:  nopw   0x0(%rax,%rax,1)
```

Display the source code listing of the mmput() function:

```
  crash> dis -s mmput
  FILE: kernel/fork.c
  LINE: 617

    612
    613   /*
    614    * Decrement the use count and release all resources for an mm.
    615    */
    616   void mmput(struct mm_struct *mm)
  * 617   {
```

```
       618           might_sleep();
       619
       620           if (atomic_dec_and_test(&mm->mm_users)) {
       621                   uprobe_clear_state(mm);
       622                   exit_aio(mm);
       623                   ksm_exit(mm);
       624                   khugepaged_exit(mm); /* must run before exit_mmap */
       625                   exit_mmap(mm);
       626                   set_mm_exe_file(mm, NULL);
       627                   if (!list_empty(&mm->mmlist)) {
       628                           spin_lock(&mmlist_lock);
       629                           list_del(&mm->mmlist);
       630                           spin_unlock(&mmlist_lock);
       631                   }
       632                   if (mm->binfmt)
       633                           module_put(mm->binfmt->module);
       634                   mmdrop(mm);
       635           }
       636   }
```

The disassembly of dentry_kill() shows an indirect call to a function whose address is contained within a register. Display the source code associated with the indirect function call:

```
crash> dis dentry_kill
...
0xffffffff811dcfb4 <dentry_kill+324>:   callq  *%rax
...
crash> dis -s 0xffffffff811dcfb4
FILE: fs/dcache.c
LINE: 276

  271                   spin_unlock(&dentry->d_lock);
  272                   spin_unlock(&inode->i_lock);
  273                   if (!inode->i_nlink)
  274                           fsnotify_inoderemove(inode);
  275                   if (dentry->d_op && dentry->d_op->d_iput)
* 276                           dentry->d_op->d_iput(dentry, inode);
  277                   else
  278                           iput(inode);
  279           } else {
  280                   spin_unlock(&dentry->d_lock);
  281           }
  282   }
```

```
NAME
  kmem - kernel memory

SYNOPSIS
  kmem [-f|-F|-c|-C|-i|-v|-V|-n|-z|-o|-h] [-p | -m member[,member]]
       [[-s|-S|-S=cpu[s]|-r] [slab] [-I slab[,slab]]] [-g [flags]] [[-P] address]]

DESCRIPTION
  This command displays information about the use of kernel memory.

        -f  displays the contents of the system free memory headers.
            also verifies that the page count equals nr_free_pages.
        -F  same as -f, but also dumps all pages linked to that header.
        -c  walks through the page_hash_table and verifies page_cache_size.
        -C  same as -c, but also dumps all pages in the page_hash_table.
        -i  displays general memory usage information
        -v  displays the mapped virtual memory regions allocated by vmalloc().
        -V  displays the kernel vm_stat table if it exists, or in more recent
            kernels, the vm_zone_stat, vm_node_stat and vm_numa_stat tables,
            the cumulative page_states counter values if they exist, and/or
            the cumulative, vm_event_states counter values if they exist.
        -n  display memory node, memory section, memory block data and state;
            the state of each memory section is shown as the following flags:
              "P": SECTION_MARKED_PRESENT
              "M": SECTION_HAS_MEM_MAP
              "O": SECTION_IS_ONLINE
```

```
                "E": SECTION_IS_EARLY
                "D": SECTION_TAINT_ZONE_DEVICE
        -z  displays per-zone memory statistics.
        -o  displays each cpu's offset value that is added to per-cpu symbol
            values to translate them into kernel virtual addresses.
        -h  display the address of hugepage hstate array entries, along with
            their hugepage size, total and free counts, and name.
        -p  displays basic information about each page structure in the system
            mem_map[] array, made up of the page struct address, its associated
            physical address, the page.mapping, page.index, page._count and
            page.flags fields.
  -m member  similar to -p, but displays page structure contents specified by
            a comma-separated list of one or more struct page members.  The
            "flags" member will always be expressed in hexadecimal format, and
            the "_count" and "_mapcount" members will always be expressed
            in decimal format.  Otherwise, all other members will be displayed
            in hexadecimal format unless the output radix is 10 and the member
            is a signed/unsigned integer.  Members that are data structures may
            be specified either by the data structure's member name, or expanded
            to specify a member of the data structure.  For example, "-m lru"
            refers to a list_head data structure, and both the list_head.next
            and list_head.prev pointer values will be displayed, whereas if
            "-m lru.next" is specified, just the list_head.next value will
            be displayed.
        -s  displays basic kmalloc() slab data.
        -S  displays all kmalloc() slab data, including all slab objects,
            and whether each object is in use or is free.  If CONFIG_SLUB,
            slab data for each per-cpu slab is displayed, along with the
            address of each kmem_cache_node, its count of full and partial
            slabs, and a list of all tracked slabs.
            Note: one can specify the per-cpu slab data to be displayed;
            the cpu[s] can be given as "1,3,5", "1-3", "1,3,5-7,10",
            "all", or "a" (shortcut for "all").
        -r  displays the accumulated basic kmalloc() slab data of each
            root slab cache and its children.  The kernel must contain the
            "slab_root_caches" list_head. (currently only available if
            CONFIG_SLUB)
      slab  when used with -s, -S or -r, limits the command to only the slab
            cache of name "slab".  If the slab argument is "list", then
            all slab cache names and addresses are listed.
  -I slab  when used with -s, -S or -r, one or more slab cache names in a
            comma-separated list may be specified as slab caches to ignore.
        -g  displays the enumerator value of all bits in the page structure's
            "flags" field.
     flags  when used with -g, translates all bits in this hexadecimal page
            structure flags value into its enumerator values.
        -P  declares that the following address argument is a physical address.
   address  when used without any flag, the address can be a kernel virtual,
            or physical address; a search is made through the symbol table,
            the kmalloc() slab subsystem, the free list, the page_hash_table,
            the vmalloc() region subsystem, the current set of task_structs
            and kernel stacks, and the mem_map array.  If found in any of
            those areas, the information will be dumped in the same manner as
            if the location-specific flags were used; if contained within a
            current task_struct or kernel stack, that task's context will be
            displayed.
   address  when used with -s or -S, searches the kmalloc() slab subsystem
            for the slab containing of this virtual address, showing whether
            it is in use or free.
   address  when used with -f, the address can be either a page pointer,
            a physical address, or a kernel virtual address; the free_area
            header containing the page (if any) is displayed.
   address  when used with -p, the address can be either a page pointer, a
            physical address, or a kernel virtual address; its basic mem_map
            page information is displayed.
   address  when used with -m, the address can be either a page pointer, a
            physical address, or a kernel virtual address; the specified
            members of the associated page struct are displayed.
   address  when used with -c, the address must be a page pointer address;
            the page_hash_table entry containing the page is displayed.
   address  when used with -l, the address must be a page pointer address;
            the page address is displayed if it is contained with the list.
   address  when used with -v, the address can be a mapped kernel virtual
```

```
              address or physical address; the mapped region containing the
              address is displayed.

    All address arguments above must be expressed in hexadecimal format.

EXAMPLES
    Display memory usage information:

      crash> kmem -i
                       PAGES        TOTAL      PERCENTAGE
          TOTAL MEM  1974231       7.5 GB         ----
               FREE   208962     816.3 MB   10% of TOTAL MEM
               USED  1765269       6.7 GB   89% of TOTAL MEM
             SHARED   365066       1.4 GB   18% of TOTAL MEM
            BUFFERS   111376     435.1 MB    5% of TOTAL MEM
             CACHED  1276196       4.9 GB   64% of TOTAL MEM
               SLAB   120410     470.4 MB    6% of TOTAL MEM

         TOTAL HUGE   524288         2 GB         ----
          HUGE FREE   524288         2 GB  100% of TOTAL HUGE

         TOTAL SWAP  2498559       9.5 GB         ----
          SWAP USED    81978     320.2 MB    3% of TOTAL SWAP
          SWAP FREE  2416581       9.2 GB   96% of TOTAL SWAP

       COMMIT LIMIT  3485674      13.3 GB         ----
          COMMITTED   850651       3.2 GB   24% of TOTAL LIMIT

    Display and verify free memory data:

      crash> kmem -f
      NODE
        0
      ZONE  NAME        SIZE    FREE  MEM_MAP   START_PADDR  START_MAPNR
        0   DMA         4096    3372  c4000040       0            0
      AREA  SIZE  FREE_AREA_STRUCT  BLOCKS  PAGES
        0     4k      c02eb004          2      2
        1     8k      c02eb010          3      6
        2    16k      c02eb01c          5     20
        3    32k      c02eb028          4     32
        4    64k      c02eb034          5     80
        5   128k      c02eb040          3     96
        6   256k      c02eb04c          3    192
        7   512k      c02eb058          1    128
        8  1024k      c02eb064          1    256
        9  2048k      c02eb070          5   2560

      ZONE  NAME        SIZE    FREE  MEM_MAP   START_PADDR  START_MAPNR
        1   Normal    225280  202269  c4044040    1000000        4096
      AREA  SIZE  FREE_AREA_STRUCT  BLOCKS  PAGES
        0     4k      c02eb0b8          1      1
        1     8k      c02eb0c4          2      4
        2    16k      c02eb0d0          0      0
        3    32k      c02eb0dc          1      8
        4    64k      c02eb0e8          1     16
        5   128k      c02eb0f4          0      0
        6   256k      c02eb100          0      0
        7   512k      c02eb10c          0      0
        8  1024k      c02eb118          0      0
        9  2048k      c02eb124        395 202240

      ZONE  NAME        SIZE    FREE  MEM_MAP   START_PADDR  START_MAPNR
        2   HighMem   819200  748686  c4ee0040    38000000      229376
      AREA  SIZE  FREE_AREA_STRUCT  BLOCKS  PAGES
        0     4k      c02eb16c         10     10
        1     8k      c02eb178          2      4
        2    16k      c02eb184          0      0
        3    32k      c02eb190          2     16
        4    64k      c02eb19c          1     16
        5   128k      c02eb1a8          1     32
        6   256k      c02eb1b4          1     64
        7   512k      c02eb1c0          0      0
        8  1024k      c02eb1cc          0      0
```

```
        9  2048k      c02eb1d8        1462 748544

      nr_free_pages: 954327  (verified)
```

Dump all the base addresses of each free memory area from above:

```
      crash> kmem -F
      NODE
        0
      ZONE  NAME        SIZE    FREE  MEM_MAP   START_PADDR  START_MAPNR
        0   DMA         4096    3372  c4000040       0            0
      AREA  SIZE  FREE_AREA_STRUCT
        0     4k      c02eb004
      c400ded8
      c4042528
      AREA  SIZE  FREE_AREA_STRUCT
        1     8k      c02eb010
      c400de50
      c400cee8
      c40424a0
      AREA  SIZE  FREE_AREA_STRUCT
        2    16k      c02eb01c
      c400dd40
      c400cf70
      c40425b0
      c400f7d0
      c40028a0
      AREA  SIZE  FREE_AREA_STRUCT
        3    32k      c02eb028
      c4042280
      c400f8e0
      c4002680
      c4000260
      AREA  SIZE  FREE_AREA_STRUCT
        4    64k      c02eb034
      c400d080
      c4041e40
      ...
```

Dump the mem_map[] array:

```
      crash> kmem -p
        PAGE    PHYSICAL   MAPPING    INDEX CNT FLAGS
      f5c51200     10000         0        0  1 80 slab
      f5c51220     11000         0        0  1 80 slab
      f5c51240     12000         0        0  1 80 slab
      f5c51260     13000         0        0  1 80 slab
      f5c51280     14000         0        0  1 80 slab
      f5c512a0     15000         0        0  1 80 slab
      f5c512c0     16000         0        0  1 80 slab
      f5c512e0     17000         0        0  1 80 slab
      f5c51300     18000         0        0  1 80 slab
      f5c51320     19000         0        0  1 80 slab
      f5c51340     1a000         0        0  1 80 slab
      f5c51360     1b000         0        0  1 80 slab
      f5c51380     1c000  e6c6a754     13b67  2 868 uptodate,lru,active,private
      f5c513a0     1d000         0        0  1 80 slab
      f5c513c0     1e000         0        0  1 80 slab
      f5c513e0     1f000         0        0  1 80 slab
      f5c51400     20000  e6c6a754     13bbb  2 868 uptodate,lru,active,private
      f5c51420     21000         0        0  1 80 slab
      f5c51440     22000         0        0  1 80 slab
      ...
```

Display the "page.lru" list_head structure member in each page:

```
      crash> kmem -m lru
           PAGE         lru
      ffffea0000000000  0000000000000000,0000000000000000
      ffffea0000000040  ffffea0000000060,ffffea0000000060
      ffffea0000000080  ffffea00000000a0,ffffea00000000a0
      ffffea00000000c0  ffffea00000000e0,ffffea00000000e0
      ffffea0000000100  ffffea0000000120,ffffea0000000120
```

```
  ffffea0000000140  ffffea0000000160,ffffea0000000160
  ffffea0000000180  ffffea00000001a0,ffffea00000001a0
  ffffea00000001c0  ffffea00000001e0,ffffea00000001e0
  ffffea0000000200  ffffea0000000220,ffffea0000000220
  ffffea0000000240  ffffea0000000260,ffffea0000000260
  ffffea0000000280  ffffea00000002a0,ffffea00000002a0
  ffffea00000002c0  ffffea00000002e0,ffffea00000002e0
  ffffea0000000300  ffffea0000000320,ffffea0000000320
  ffffea0000000340  ffffea0000000360,ffffea0000000360
  ffffea0000000380  ffffea00000003a0,ffffea00000003a0
  ffffea00000003c0  ffffea00000003e0,ffffea00000003e0
  ffffea0000000400  ffff88021e5e41e8,ffffea0000002020
  ffffea0000000440  dead000000100100,dead000000200200
  ffffea0000000480  dead000000100100,dead000000200200
  ffffea00000004c0  dead000000100100,dead000000200200
  ...
```

Find the two pages that link to the page at ffffea0001dafb20
via their page.lru list_head's next and prev pointers:

```
  crash> kmem -m lru | grep ffffea0001dafb20
  ffffea000006b500  ffffea0001dafb20,ffffea0001eb4520
  ffffea0000127d80  ffffea000152b620,ffffea0001dafb20
```

Find all of the combined slab/page structures that are used by
the kmalloc-8192 slab cache:

```
  crash> kmem -s kmalloc-8192
  CACHE             OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE  NAME
  ffff880215802e00     8192         65        80     20    32k  kmalloc-8192
  crash> kmem -m slab_cache | grep ffff880215802e00
  ffffea0004117800  ffff880215802e00
  ffffea00041ca600  ffff880215802e00
  ffffea00044ab200  ffff880215802e00
  ffffea0004524000  ffff880215802e00
  ffffea0004591600  ffff880215802e00
  ffffea00047eac00  ffff880215802e00
  ffffea0004875800  ffff880215802e00
  ffffea0008357a00  ffff880215802e00
  ffffea0008362a00  ffff880215802e00
  ffffea00083b9400  ffff880215802e00
  ffffea00083c1000  ffff880215802e00
  ffffea00083c1e00  ffff880215802e00
  ffffea00083c2000  ffff880215802e00
  ffffea00083c2a00  ffff880215802e00
  ffffea00083d2000  ffff880215802e00
  ffffea00083d3e00  ffff880215802e00
  ffffea0008407c00  ffff880215802e00
  ffffea000848ce00  ffff880215802e00
  ffffea0008491800  ffff880215802e00
  ffffea00084bf800  ffff880215802e00
```

Use the commands above with a page pointer or a physical address argument:

```
  crash> kmem -f c40425b0
  NODE
    0
  ZONE  NAME        SIZE    FREE  MEM_MAP   START_PADDR  START_MAPNR
    0   DMA         4096    3372  c4000040       0            0
  AREA  SIZE  FREE_AREA_STRUCT
    2    16k      c02eb01c
  c40425b0  (c40425b0 is 1st of 4 pages)

  crash> kmem -p c25a9c00
    PAGE     PHYSICAL   MAPPING    INDEX CNT FLAGS
  c25a9c00    1fe0000  f429d2e4   21fe3eb  2 800828 uptodate,lru,private

  crash> kmem -p 1fe0000
    PAGE     PHYSICAL   MAPPING    INDEX CNT FLAGS
  c25a9c00    1fe0000  f429d2e4   21fe3eb  2 800828 uptodate,lru,private
```

Display the mapped memory regions allocated by vmalloc():

```
crash> kmem -v
VMAP_AREA  VM_STRUCT     ADDRESS RANGE        SIZE
f7048e00   f7048e40   f7dfe000 - f7e00000     8192
f7048ec0   f7048f00   f7e00000 - f7e05000    20480
f7151fc0   f7159540   f7e06000 - f7e08000     8192
f704da80   f704dac0   f7e0a000 - f7e0c000     8192
f704d980   f704d9c0   f7e0e000 - f7e10000     8192
f724f1c0   f724f200   f7e12000 - f7e14000     8192
f704d840   f704d880   f7e14000 - f7e17000    12288
f704d400   f704d440   f7e18000 - f7e1d000    20480
f73f5840   f73f5880   f7e1e000 - f7e2a000    49152
f6334480   f63344c0   f7e2c000 - f7e2e000     8192
f635d600   f635d640   f7e4a000 - f7e5b000    69632
f41b4700   f5771a40   f7e6e000 - f7e70000     8192
f622f6c0   f622f700   f7e71000 - f7e79000    32768
f63a9f00   f63a9f40   f7e84000 - f7e87000    12288
f63a9d00   f63a9d40   f7e8f000 - f7e91000     8192
f5546480   f39db800   f7eb8000 - f7ec2000    40960
f5ce9640   f5777e80   f7ec6000 - f7ed1000    45056
f63a9b00   f63a9b40   f7ed1000 - f7efd000   180224
f63a9800   f63a9840   f7f1d000 - f7f26000    36864
f63a9640   f63a9880   f7f43000 - f7f52000    61440
f5771f00   f4183840   f7f53000 - f7f64000    69632
f5ce9a00   f30c4a00   f7fcf000 - f801e000   323584
f63a93c0   f63a9400   f805d000 - f8132000   872448
f63a91c0   f63a95c0   f814b000 - f8150000    20480
f63a9140   f63a9180   f8151000 - f8352000  2101248
f624eb00   f624eb40   f8353000 - f8355000     8192
f563eb40   f563eb80   f8356000 - f835e000    32768
f63d5ec0   f63d5f00   f8360000 - f8371000    69632
f63d5cc0   f6287b80   f83c2000 - f84c3000  1052672
...
```

Dump the virtual memory statistics:

```
crash> kmem -V
  VM_ZONE_STAT:
          NR_FREE_PAGES: 30085
 NR_ZONE_INACTIVE_ANON: 1985
   NR_ZONE_ACTIVE_ANON: 338275
 NR_ZONE_INACTIVE_FILE: 19760
   NR_ZONE_ACTIVE_FILE: 12018
   NR_ZONE_UNEVICTABLE: 0
 NR_ZONE_WRITE_PENDING: 4
              NR_MLOCK: 0
         NR_PAGETABLE: 1562
    NR_KERNEL_STACK_KB: 1728
             NR_BOUNCE: 0
     NR_FREE_CMA_PAGES: 0

  VM_NODE_STAT:
      NR_INACTIVE_ANON: 1985
        NR_ACTIVE_ANON: 338275
      NR_INACTIVE_FILE: 19760
        NR_ACTIVE_FILE: 12018
        NR_UNEVICTABLE: 0
   NR_SLAB_RECLAIMABLE: 3111
 NR_SLAB_UNRECLAIMABLE: 3039
      NR_ISOLATED_ANON: 0
      NR_ISOLATED_FILE: 0
    WORKINGSET_REFAULT: 0
   WORKINGSET_ACTIVATE: 0
WORKINGSET_NODERECLAIM: 0
        NR_ANON_MAPPED: 338089
        NR_FILE_MAPPED: 8102
         NR_FILE_PAGES: 33949
         NR_FILE_DIRTY: 4
          NR_WRITEBACK: 0
     NR_WRITEBACK_TEMP: 0
               NR_SHMEM: 2171
          NR_SHMEM_THPS: 0
      NR_SHMEM_PMDMAPPED: 0
           NR_ANON_THPS: 86
```

```
            NR_UNSTABLE_NFS: 0
            NR_VMSCAN_WRITE: 0
        NR_VMSCAN_IMMEDIATE: 0
                  NR_DIRTIED: 155
                  NR_WRITTEN: 75

      VM_NUMA_STAT:
                   NUMA_HIT: 575409
                  NUMA_MISS: 0
                NUMA_FOREIGN: 0
         NUMA_INTERLEAVE_HIT: 12930
                  NUMA_LOCAL: 575409
                  NUMA_OTHER: 0

      VM_EVENT_STATES:
                             PGPGIN: 282492
                            PGPGOUT: 6773
                             PSWPIN: 0
                            PSWPOUT: 0
                       PGALLOC_DMA: 0
                     PGALLOC_DMA32: 693092
                    PGALLOC_NORMAL: 0
  ...
```

Display hugepage hstate information:

```
  crash> kmem -h
       HSTATE        SIZE    FREE   TOTAL  NAME
  ffffffff81f7a800    2MB      10      64  hugepages-2048kB
```

Determine (and verify) the page cache size:

```
  crash> kmem -c
  page_cache_size: 18431 (verified)
```

Dump all pages in the page_hash_table:

```
  crash> kmem -C
  page_hash_table[0]
  c0325b40
  c03a0598
  c03b4070
  c0364c28
  c0357690
  c02ef338
  c02d7c60
  c02c11e0
  c02a3d70
  page_hash_table[1]
  c0394ce8
  c03c4218
  c03b4048
  c0364c00
  c0357668
  c02d6e50
  c02d7dc8
  c02c0cb8
  c02db630
  c02ebad0
  page_hash_table[2]
  c037e808
  c034e248
  c03b4020
  c02ec868
  c03baa60
  ...
  page_hash_table[2047]
  c033a798
  c0390b48
  c03b4098
  c0364890
  c03576b8
  c02d2c38
```

```
  c02d7c88
  c02de5d8

  page_cache_size: 18437 (verified)

Find the page_hash_table entry containing page c03576b8:

  crash> kmem -c c03576b8
  page_hash_table[2047]
  c03576b8

Display kmalloc() slab data:

  crash> kmem -s
  CACHE     OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE  NAME
  c02eadc0      232         58        68      4     4k  kmem_cache
  f79c2888      128          0         0      0     4k  ip_vs_conn
  f79c2970       96          0         0      0     4k  tcp_tw_bucket
  f79c2a58       32         12       565      5     4k  tcp_bind_bucket
  f79c2b40       64          0        59      1     4k  tcp_open_request
  f79c2c28       64          1        59      1     4k  inet_peer_cache
  f79c2d10       32         11       339      3     4k  ip_fib_hash
  f79c2df8      160          8       120      5     4k  ip_dst_cache
  f79c2ee0      128          1        30      1     4k  arp_cache
  c8402970       96      30208     37800    945     4k  blkdev_requests
  c8402a58      384          0         0      0     4k  nfs_read_data
  c8402b40      384          0         0      0     4k  nfs_write_data
  c8402c28       96          0         0      0     4k  nfs_page
  c8402d10       20          0         0      0     4k  dnotify cache
  c8402df8       92          3       336      8     4k  file lock cache
  c8402ee0       16          0         0      0     4k  fasync cache
  c84027a0       32          3       339      3     4k  uid_cache
  c84026b8      160        320       624     26     4k  skbuff_head_cache
  c84025d0      832         32       180     20     8k  sock
  c84024e8      132          0       203      7     4k  sigqueue
  c8402400       64         19       472      8     4k  cdev_cache
  c8402318       64          8       236      4     4k  bdev_cache
  c8402230       96         11       120      3     4k  mnt_cache
  c8402148      480        817       848    106     4k  inode_cache
  c8402060      128       1352      1470     49     4k  dentry_cache
  c8403ee0       96        244       440     11     4k  filp
  c8403df8     4096          0        12     12     4k  names_cache
  c8403d10       96      14936     16000    400     4k  buffer_head
  c8403c28      128         25       240      8     4k  mm_struct
  c8403b40       64        393      1298     22     4k  vm_area_struct
  c8403a58       64         30       472      8     4k  fs_cache
  c8403970      416         30       135     15     4k  files_cache
  c8403888     1312         32        99     33     4k  signal_act
  c84037a0   131072          0         0      0   128k  size-131072(DMA)
  c84036b8   131072          1         1      1   128k  size-131072
  c84035d0    65536          0         0      0    64k  size-65536(DMA)
  c84034e8    65536          0         0      0    64k  size-65536
  c8403400    32768          0         0      0    32k  size-32768(DMA)
  c8403318    32768          0         1      1    32k  size-32768
  c8403230    16384          0         0      0    16k  size-16384(DMA)
  c8403148    16384          0         0      0    16k  size-16384
  c8403060     8192          0         0      0     8k  size-8192(DMA)
  c8401ee0     8192          1         2      2     8k  size-8192
  c8401df8     4096          0         0      0     4k  size-4096(DMA)
  c8401d10     4096         30        30     30     4k  size-4096
  c8401c28     2048          0         0      0     4k  size-2048(DMA)
  c8401b40     2048         37       132     66     4k  size-2048
  c8401a58     1024          0         0      0     4k  size-1024(DMA)
  c8401970     1024        301       328     82     4k  size-1024
  c8401888      512          0         0      0     4k  size-512(DMA)
  c84017a0      512        141       168     21     4k  size-512
  c84016b8      256          0         0      0     4k  size-256(DMA)
  c84015d0      256         80       435     29     4k  size-256
  c84014e8      128          0         0      0     4k  size-128(DMA)
  c8401400      128        508       840     28     4k  size-128
  c8401318       64          0         0      0     4k  size-64(DMA)
  c8401230       64        978      1357     23     4k  size-64
  c8401148       32          0         0      0     4k  size-32(DMA)
```

```
  c8401060       32       1244      1808     16     4k  size-32
```

Display all slab data in the "arp_cache" cache:

```
  crash> kmem -S arp_cache
  CACHE     OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE  NAME
  f79c2ee0      128          1        30      1     4k  arp_cache
  SLAB      MEMORY    TOTAL  ALLOCATED  FREE
  f729d000  f729d0a0     30          1    29
  FREE / [ALLOCATED]
     f729d0a0  (cpu 7 cache)
     f729d120  (cpu 7 cache)
     f729d1a0  (cpu 7 cache)
     f729d220  (cpu 7 cache)
     f729d2a0  (cpu 7 cache)
     f729d320  (cpu 7 cache)
     f729d3a0  (cpu 7 cache)
     f729d420  (cpu 7 cache)
     f729d4a0  (cpu 7 cache)
     f729d520  (cpu 7 cache)
     f729d5a0  (cpu 7 cache)
     f729d620  (cpu 7 cache)
     f729d6a0  (cpu 7 cache)
     f729d720  (cpu 7 cache)
     f729d7a0  (cpu 7 cache)
     f729d820  (cpu 7 cache)
     f729d8a0  (cpu 7 cache)
     f729d920  (cpu 7 cache)
     f729d9a0  (cpu 7 cache)
     f729da20  (cpu 7 cache)
     f729daa0  (cpu 7 cache)
     f729db20  (cpu 7 cache)
     f729dba0  (cpu 7 cache)
     f729dc20  (cpu 7 cache)
     f729dca0  (cpu 7 cache)
     f729dd20  (cpu 7 cache)
     f729dda0  (cpu 7 cache)
     f729de20  (cpu 7 cache)
     f729dea0  (cpu 3 cache)
    [f729df20]
```

Search the kmalloc() slab subsystem for address c3fbdb60:

```
  crash> kmem -s c3fbdb60
  CACHE     OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE  NAME
  c8402970       96      30208     37800    945     4k  blkdev_requests
  SLAB      MEMORY    TOTAL  ALLOCATED  FREE
  c3fbd020  c3fbd0e0     40         40     0
  FREE / [ALLOCATED]
    [c3fbdb60]
```

Make a generic search (no flags) for the same address c3fbdb60:

```
  crash> kmem c3fbdb60
  CACHE     OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE  NAME
  c8402970       96      30208     37800    945     4k  blkdev_requests
  SLAB      MEMORY    TOTAL  ALLOCATED  FREE
  c3fbd020  c3fbd0e0     40         40     0
  FREE / [ALLOCATED]
    [c3fbdb60]

      PAGE     PHYSICAL   MAPPING    INDEX CNT FLAGS
  c410ee74    3fbd000         0         0  1 slab
```

Display memory node data (if supported):

```
  crash> kmem -n
  NODE    SIZE      PGLIST_DATA       BOOTMEM_DATA       NODE_ZONES
    0    262095   ffff88003d52a000        ----        ffff88003d52a000
                                                      ffff88003d52a740
                                                      ffff88003d52ae80
                                                      ffff88003d52b5c0
        MEM_MAP          START_PADDR    START_MAPNR
```

```
      ffffea0000000040        1000             1

      ZONE  NAME         SIZE       MEM_MAP      START_PADDR  START_MAPNR
        0   DMA          4095  ffffea0000000040         1000            1
        1   DMA32      258000  ffffea0000040000      1000000         4096
        2   Normal          0                 0            0            0
        3   Movable         0                 0            0            0

      --------------------------------------------------------------------

      NR       SECTION        CODED_MEM_MAP        MEM_MAP       STATE PFN
       0  ffff88003d4d9000  ffffea0000000000  ffffea0000000000   PM   0
       1  ffff88003d4d9020  ffffea0000000000  ffffea0000200000   PM   32768
       2  ffff88003d4d9040  ffffea0000000000  ffffea0000400000   PM   65536
       3  ffff88003d4d9060  ffffea0000000000  ffffea0000600000   PM   98304
       4  ffff88003d4d9080  ffffea0000000000  ffffea0000800000   PM   131072
       5  ffff88003d4d90a0  ffffea0000000000  ffffea0000a00000   PM   163840
       6  ffff88003d4d90c0  ffffea0000000000  ffffea0000c00000   PM   196608
       7  ffff88003d4d90e0  ffffea0000000000  ffffea0000e00000   PM   229376

         MEM_BLOCK        NAME     PHYSICAL RANGE      STATE   START_SECTION_NO
       ffff88003a707c00  memory0          0 -  7ffffff ONLINE  0
       ffff88003a6e0000  memory1    8000000 -  fffffff ONLINE  1
       ffff88003a6e1000  memory2   10000000 - 17ffffff ONLINE  2
       ffff88003a6e1400  memory3   18000000 - 1fffffff ONLINE  3
       ffff88003a6e1800  memory4   20000000 - 27ffffff ONLINE  4
       ffff88003a6e0400  memory5   28000000 - 2fffffff ONLINE  5
       ffff88003a6e0800  memory6   30000000 - 37ffffff ONLINE  6
       ffff88003a6e0c00  memory7   38000000 - 3fffffff ONLINE  7
```

Translate a page structure's flags field contents:

```
  crash> kmem -g 4080
  FLAGS: 4080
    PAGE-FLAG        BIT  VALUE
    PG_slab            7  0000080
    PG_head           14  0004000
  crash>
```

NAME

ptov - physical to virtual
per-cpu to virtual

SYNOPSIS

```
ptov [address | offset:cpuspec]
```

DESCRIPTION

This command translates a hexadecimal physical address into a kernel virtual address. Alternatively, a hexadecimal per-cpu offset and cpu specifier will be translated into kernel virtual addresses for each cpu specified.

```
        address  a physical address
offset:cpuspec  a per-cpu offset with a CPU specifier:
                  :             CPU of the currently selected task.
                  :a[ll]        all CPUs.
                  :#[-#][,...]  CPU list(s), e.g. "1,3,5", "1-3",
                                or "1,3,5-7,10".
```

EXAMPLES

Translate physical address 56e000 into a kernel virtual address:

```
  crash> ptov 56e000
  VIRTUAL           PHYSICAL
  ffff88000056e000  56e000
```

Translate per-cpu offset b0c0 into a kernel virtual address for all cpus:

```
  crash> ptov b0c0:a
  PER-CPU OFFSET: b0c0
    CPU    VIRTUAL
```

```
        [0]  ffff88021e20b0c0
        [1]  ffff88021e24b0c0
        [2]  ffff88021e28b0c0
        [3]  ffff88021e2cb0c0


NAME
  sys - system data

SYNOPSIS
  sys [-c [name|number]] [-t] [-i] config

DESCRIPTION
  This command displays system-specific data.  If no arguments are entered,
  the same system data shown during crash invocation is shown.

    -c [name|number]  If no name or number argument is entered, dump all
                      sys_call_table entries.  If a name string is entered,
                      search the table for all entries containing the string.
                      If a number is entered, the table entry associated with
                      that number is displayed.  If the current output radix
                      has been set to 16, the system call numbers will be
                      displayed in hexadecimal.
    config            If the kernel was configured with CONFIG_IKCONFIG, then
                      dump the in-kernel configuration data.
    -t                Display kernel taint information.  If the "tainted_mask"
                      symbol exists, show its hexadecimal value and translate
                      each bit set to the symbolic letter of the taint type.
                      On older kernels with the "tainted" symbol, only its
                      hexadecimal value is shown.  The relevant kernel sources
                      should be consulted for the meaning of the letter(s) or
                      hexadecimal bit value(s).
    -panic            Panic a live system.  Requires write permission to
                      /dev/mem.  Results in the crash context causing an
                      "Attempted to kill the idle task!" panic.  (The dump
                      will indicate that the crash context has a PID of 0).
    -i                Dump the DMI string data if available in the kernel.

EXAMPLES
  Display essential system information:

    crash> sys
          KERNEL: vmlinux.4
        DUMPFILE: lcore.cr.4
            CPUS: 4
            DATE: Mon Oct 11 18:48:55 1999
          UPTIME: 10 days, 14:14:39
    LOAD AVERAGE: 0.74, 0.23, 0.08
           TASKS: 77
        NODENAME: test.mclinux.com
         RELEASE: 2.2.5-15smp
         VERSION: #24 SMP Mon Oct 11 17:41:40 CDT 1999
         MACHINE: i686  (500 MHz)
          MEMORY: 1 GB

  Dump the system configuration data (if CONFIG_IKCONFIG):

    crash> sys config
    #
    # Automatically generated make config: don't edit
    # Linux kernel version: 2.6.16
    # Mon Apr 10 07:58:06 2006
    #
    CONFIG_X86_64=y
    CONFIG_64BIT=y
    CONFIG_X86=y
    CONFIG_SEMAPHORE_SLEEPERS=y
    CONFIG_MMU=y
    CONFIG_RWSEM_GENERIC_SPINLOCK=y
    CONFIG_GENERIC_CALIBRATE_DELAY=y
    CONFIG_X86_CMPXCHG=y
    CONFIG_EARLY_PRINTK=y
    CONFIG_GENERIC_ISA_DMA=y
```

```
      CONFIG_GENERIC_IOMAP=y
      CONFIG_ARCH_MAY_HAVE_PC_FDC=y
      CONFIG_DMI=y
      ...

  Display the kernel taint information, in this case where both the
  TAINT_WARN and TAINT_PROPRIETARY_MODULE bits have been set:

    crash> sys -t
    TAINTED_MASK: 201  PW

  Dump the system call table:

    crash> sys -c
    NUM  SYSTEM CALL                FILE AND LINE NUMBER
      0  sys_ni_syscall             ../kernel/sys.c: 48
      1  sys_exit                   ../kernel/exit.c: 404
      2  sys_fork                   ../arch/i386/kernel/process.c: 771
      3  sys_read                   ../fs/read_write.c: 117
      4  sys_write                  ../fs/read_write.c: 146
      5  sys_open                   ../fs/open.c: 754
      6  sys_close                  ../fs/open.c: 839
      7  sys_waitpid                ../kernel/exit.c: 503
      8  sys_creat                  ../fs/open.c: 789
      9  sys_link                   ../fs/namei.c: 1213
     10  sys_unlink                 ../fs/namei.c: 1074
     11  sys_execve                 ../arch/i386/kernel/process.c: 806
    ...

  Find the system call number of the select system call:

    crash> sys -c select
    NUM  SYSTEM CALL                FILE AND LINE NUMBER
     65  sys_select                 ../fs/select.c: 259

    If the current output radix has been set to 16, the system call numbers
    will be displayed in hexadecimal.

  Dump the DMI string data:

    crash> sys -i
            DMI_BIOS_VENDOR: LENOVO
           DMI_BIOS_VERSION: G4ET37WW (1.12 )
              DMI_BIOS_DATE: 05/29/2012
            DMI_SYS_VENDOR: LENOVO
           DMI_PRODUCT_NAME: 2429BQ1
        DMI_PRODUCT_VERSION: ThinkPad T530
         DMI_PRODUCT_SERIAL: R9R91HZ
           DMI_PRODUCT_UUID: 568DFA01-5180-11CB-B851-BD06085ADDB0
           DMI_BOARD_VENDOR: LENOVO
             DMI_BOARD_NAME: 2429BQ1
          DMI_BOARD_VERSION: Not Available
           DMI_BOARD_SERIAL: 1ZLV127F17M
       DMI_BOARD_ASSET_TAG: Not Available
         DMI_CHASSIS_VENDOR: LENOVO
           DMI_CHASSIS_TYPE: 10
       DMI_CHASSIS_VERSION: Not Available
          DMI_CHASSIS_SERIAL: R9R91HZ
      DMI_CHASSIS_ASSET_TAG: RH0004111


NAME
  q - exit this session

SYNOPSIS
  q

DESCRIPTION
  Bail out of the current crash session.

NOTE
  This command is equivalent to the "exit" command.
```

```
NAME
  eval - evaluate

SYNOPSIS
  eval [-b][-l] (expression) | value

DESCRIPTION
  This command evaluates an expression or numeric value, and displays its
  result in hexadecimal, decimal, octal and binary. If the resultant value
  is an integral number of gigabytes, megabytes, or kilobytes, a short-hand
  translation of the number will also be shown next to the hexadecimal
  value.  If the most significant bit is set, the decimal display will show
  both unsigned and signed (negative) values.  Expressions must of the format
  (x operator y), where "x" and "y" may be either numeric values or
  symbols.  The list of operators are:

                     +  -  &  |  ^  *  %  /  <<  >>

  Enclosing the expression within parentheses is optional except when the
  "|", "<<" or ">>" operators are used.  The single "value" argument may
  be a number or symbol.  Number arguments must be hexadecimal or decimal.
  A leading "0x" identifies a number as hexadecimal, but is not required
  when obvious.  Numbers may be followed by the letters "k" or "K", "m"
  or "M", and "g" or "G", which multiplies the value by a factor of 1024,
  1 megabyte or 1 gigabyte, respectively.  Numeric arguments may be preceded
  by the one's complement operator ~.

    -b  Indicate which bit positions in the resultant value are set.
    -l  Numeric arguments are presumed to be 64-bit values, and the result
        will be expressed as a 64-bit value. (ignored on 64-bit processors)
        However, if either operand or the resultant value are 64-bit values,
        then the result will be also be expressed as a 64-bit value.

 The -b and -l options must precede the expression or value arguments.

EXAMPLES
   crash> eval 128m
   hexadecimal: 8000000  (128MB)
       decimal: 134217728
         octal: 1000000000
        binary: 00001000000000000000000000000000

   crash> eval 128 * 1m
   hexadecimal: 8000000  (128MB)
       decimal: 134217728
         octal: 1000000000
        binary: 00001000000000000000000000000000

   crash> eval (1 << 27)
   hexadecimal: 8000000  (128MB)
       decimal: 134217728
         octal: 1000000000
        binary: 00001000000000000000000000000000

   crash> eval (1 << 32)
   hexadecimal: 100000000  (4GB)
       decimal: 4294967296
         octal: 40000000000
        binary: 0000000000000000000000000000000100000000000000000000000000000000

   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 -lb 64g
   hexadecimal: 1000000000  (64GB)
       decimal: 68719476736
         octal: 1000000000000
        binary: 0000000000000000000000000001000000000000000000000000000000000000
```

```
       bits set: 36


NAME
  list - linked list

SYNOPSIS
  list [[-o] offset][-e end][-[s|S] struct[.member[,member] [-l offset]] -[x|d]]
       [-r|-B] [-h [-O head_offset]|-H] start

DESCRIPTION

  This command dumps the contents of a linked list.  The entries in a linked
  list are typically data structures that are tied together in one of two
  formats:

  1. A starting address points to a data structure; that structure contains
     a member that is a pointer to the next structure, and so on.  This type
     of a singly-linked list typically ends when a "next" pointer value
     contains one of the following:

       (a) a NULL pointer.
       (b) a pointer to the start address.
       (c) a pointer to the first item pointed to by the start address.
       (d) a pointer to its containing structure.

  2. Most Linux lists of data structures are doubly-linked using "list_head"
     structures that are embedded members of the data structures in the list:

       struct list_head {
           struct list_head *next, *prev;
       };

     The linked list is typically headed by an external, standalone list_head,
     which is simply initialized to point to itself, signifying that the list
     is empty:

       #define LIST_HEAD_INIT(name) { &(name), &(name) }
       #define LIST_HEAD(name) struct list_head name = LIST_HEAD_INIT(name)

     In the case of list_head-linked lists, the "list_head.next" pointer is
     the address of a list_head structure that is embedded in the next data
     structure in the list, and not the address of the next data structure
     itself.  The starting point of the list may be:

       (a) an external, standalone, LIST_HEAD().
       (b) a list_head that is embedded within a data structure of the same
           type as the whole linked list.
       (c) a list_head that is embedded within a data structure that is
           different than the type of structures in the the linked list.

     The list typically ends when the embedded "list_head.next" pointer of
     a data structure in the linked list points back to the LIST_HEAD()
     address.  However, some list_head-linked lists have no defined starting
     point, but just link back onto themselves in a circular manner.

  This command can handle both types of linked list; in both cases the list
  of addresses that are dumped are the addresses of the data structures
  themselves.

  Alternatively, the address of a list_head, or other similar list linkage
  structure whose first member points to the next linkage structure, may be
  used as the starting address.  The caveat with this type of usage is that
  the list may pass through, and display the address of, an external standalone
  list head which is not an address of a list linkage structure that is embedded
  within the data structure of interest.

  The arguments are as follows:

  [-o] offset  The offset within the structure to the "next" pointer
               (default is 0).  If non-zero, the offset may be entered
               in either of two manners:
```

1. In "structure.member" format; the "-o" is not necessary.
2. A number of bytes; the "-o" is only necessary on processors where the offset value could be misconstrued as a kernel virtual address.

-e end If the list ends in a manner unlike the typical manners that are described above, an explicit ending address value may be entered.

-s struct For each address in list, format and print as this type of structure; use the "struct.member" format in order to display a particular member of the structure. To display multiple members of a structure, use a comma-separated list of members. If any structure member contains an embedded structure or is an array, the output may be restricted to the embedded structure or an array element by expressing the struct argument as "struct.member.member" or "struct.member[index]"; embedded member specifications may extend beyond one level deep by expressing the argument as "struct.member.member.member...".

-S struct Similar to -s, but instead of parsing gdb output, member values are read directly from memory, so the command works much faster for 1-, 2-, 4-, and 8-byte members.

-O offset Only used in conjunction with -h; it specifies the offset of head node list_head embedded within a data structure which is different than the offset of list_head of other nodes embedded within a data structure.
The offset may be entered in either of the following manners:

1. in "structure.member" format.
2. a number of bytes.

-l offset Only used in conjunction with -s, if the start address argument is a pointer to an embedded list head (or any other similar list linkage structure whose first member points to the next linkage structure), the offset to the embedded member may be entered in either of the following manners:

1. in "structure.member" format.
2. a number of bytes.

-x Override the default output format with hexadecimal format.

-d Override the default output format with decimal format.

-r For a list linked with list_head structures, traverse the list in the reverse order by using the "prev" pointer instead of "next".

-B Use the algorithm from R. P. Brent to detect loops instead of using a hash table. This algorithm uses a tiny fixed amount of memory and so is especially helpful for longer lists. The output is slightly different than the normal list output as it will print the length of the loop, the start of the loop, and the first duplicate in the list.

The meaning of the "start" argument, which can be expressed symbolically, in hexadecimal format, or an expression evaluating to an address, depends upon whether the -h or -H option is pre-pended:

start The address of the first data structure in the list.

start When both the -s and -l options are used, the address of an embedded list_head or similar linkage structure whose first member points to the next linkage structure.

-H start The address of a list_head structure, typically that of an external, standalone LIST_HEAD(). The list typically ends when the embedded "list_head.next" of a data structure in the linked list points back to this "start" address.

-h start The address of a data structure which contains an embedded list_head. The list typically ends when the embedded "list_head.next" of a data structure in the linked list points back to the embedded list_head contained in the data structure whose address is this "start" argument.

WARNING

When the "-h start" option is used, it is possible that the list_head-linked list will:

```
      1. pass through an external standalone LIST_HEAD(), or
      2. pass through a list_head that is the actual starting list_head, but is
         contained within a data structure that is not the same type as all of
         the other data structures in the list.

    When that occurs, the data structure address displayed for that list_head
    will be incorrect, because the "-h start" option presumes that all
    list_head structures in the list are contained within the same type of
    data structure.  Furthermore, if the "-s struct[.member[,member]" option
    is used, it will display bogus data for that particular list_head.

    A similar issue may be encountered when the "start" address is an embedded
    list_head or similar linkage structure whose first member points to the next
    linkage structure.  When that occurs, the address of any external list head
    will not be distinguishable from the addresses that are embedded in the data
    structure of interest.  Furthermore, if the "-s" and "-l" options are used,
    it will display bogus structure data when passing through any external list
    head structure that is not embedded in the specified data structure type.

  EXAMPLES
    Note that each task_struct is linked to its parent's task_struct via the
    p_pptr member:

      crash> struct task_struct.p_pptr
      struct task_struct {
         [136] struct task_struct *p_pptr;
      }

    That being the case, given a task_struct pointer of c169a000, show its
    parental hierarchy back to the "init_task" (the "swapper" task):

      crash> list task_struct.p_pptr c169a000
      c169a000
      c0440000
      c50d0000
      c0562000
      c0d28000
      c7894000
      c6a98000
      c009a000
      c0252000

    Given that the "task_struct.p_pptr" offset is 136 bytes, the same
    result could be accomplished like so:

      crash> list 136 c169a000
      c169a000
      c0440000
      c50d0000
      c0562000
      c0d28000
      c7894000
      c6a98000
      c009a000
      c0252000

    The list of currently-registered file system types are headed up by a
    struct file_system_type pointer named "file_systems", and linked by
    the "next" field in each file_system_type structure.  The following
    sequence displays the structure address followed by the name and
    fs_flags members of each registered file system type:

      crash> p file_systems
      file_systems = $1 = (struct file_system_type *) 0xc03adc90
      crash> list file_system_type.next -s file_system_type.name,fs_flags c03adc90
      c03adc90
        name = 0xc02c05c8 "rootfs",
        fs_flags = 0x30,
      c03abf94
        name = 0xc02c0319 "bdev",
        fs_flags = 0x10,
      c03acb40
        name = 0xc02c07c4 "proc",
```

```
      fs_flags = 0x8,
    c03e9834
      name = 0xc02cfc83 "sockfs",
      fs_flags = 0x10,
    c03ab8e4
      name = 0xc02bf512 "tmpfs",
      fs_flags = 0x20,
    c03ab8c8
      name = 0xc02c3d6b "shm",
      fs_flags = 0x20,
    c03ac394
      name = 0xc02c03cf "pipefs",
      fs_flags = 0x10,
    c03ada74
      name = 0xc02c0e6b "ext2",
      fs_flags = 0x1,
    c03adc74
      name = 0xc02c0e70 "ramfs",
      fs_flags = 0x20,
    c03ade74
      name = 0xc02c0e76 "hugetlbfs",
      fs_flags = 0x20,
    c03adf8c
      name = 0xc02c0f84 "iso9660",
      fs_flags = 0x1,
    c03aec14
      name = 0xc02c0ffd "devpts",
      fs_flags = 0x8,
    c03e93f4
      name = 0xc02cf1b9 "pcihpfs",
      fs_flags = 0x28,
    e0831a14
      name = 0xe082f89f "ext3",
      fs_flags = 0x1,
    e0846af4
      name = 0xe0841ac6 "usbdevfs",
      fs_flags = 0x8,
    e0846b10
      name = 0xe0841acf "usbfs",
      fs_flags = 0x8,
    e0992370
      name = 0xe099176c "autofs",
      fs_flags = 0x0,
    e2dcc030
      name = 0xe2dc8849 "nfs",
      fs_flags = 0x48000,
```

In some kernels, the system run queue is a linked list headed up by the "runqueue_head", which is defined like so:

```
  static LIST_HEAD(runqueue_head);
```

The run queue linking is done with the "run_list" member of the task_struct:

```
  crash> struct task_struct.run_list
  struct task_struct {
      [60] struct list_head run_list;
  }
```

Therefore, to view the list of task_struct addresses in the run queue, either of the following commands will work:

```
  crash> list task_struct.run_list -H runqueue_head
  f79ac000
  f7254000
  f7004000
  crash> list 60 -H runqueue_head
  f79ac000
  f7254000
  f7004000
```

In some kernel versions, the vfsmount structures of the mounted filesystems are linked by the LIST_HEAD "vfsmntlist", which uses the

mnt_list list_head of each vfsmount structure in the list. To dump each vfsmount structure in the list, append the -s option:

```
crash> list -H vfsmntlist vfsmount.mnt_list -s vfsmount
c3fc9e60
struct vfsmount {
  mnt_hash = {
    next = 0xc3fc9e60,
    prev = 0xc3fc9e60
  },
  mnt_parent = 0xc3fc9e60,
  mnt_mountpoint = 0xc3fc5dc0,
  mnt_root = 0xc3fc5dc0,
  mnt_instances = {
    next = 0xc3f60a74,
    prev = 0xc3f60a74
  },
  mnt_sb = 0xc3f60a00,
  mnt_mounts = {
    next = 0xf7445e08,
    prev = 0xf7445f88
  },
  mnt_child = {
    next = 0xc3fc9e88,
    prev = 0xc3fc9e88
  },
  mnt_count = {
    counter = 209
  },
  mnt_flags = 0,
  mnt_devname = 0xc8465b20 "/dev/root",
  mnt_list = {
    next = 0xf7445f9c,
    prev = 0xc02eb828
  },
  mnt_owner = 0
}
f7445f60
struct vfsmount {
...
```

The task_struct of every task in the system is linked into a circular list by its embedded "tasks" list_head. Show the task_struct addresses and the pids of all tasks in the system using "-h" option, starting with the task_struct at ffff88012b98e040:

```
crash> list task_struct.tasks -s task_struct.pid -h ffff88012b98e040
ffff88012b98e040
  pid = 14187
ffff8801277be0c0
  pid = 14248
ffffffff81a2d020
  pid = 0
ffff88012d7dd4c0
  pid = 1
ffff88012d7dca80
  pid = 2
ffff88012d7dc040
  pid = 3
ffff88012d7e9500
  pid = 4
...
ffff88012961a100
  pid = 14101
ffff880129017580
  pid = 14134
ffff8801269ed540
  pid = 14135
ffff880128256080
  pid = 14138
ffff88012b8f4100
  pid = 14183
```

Similar to the above, display the embedded sched_entity structure's on_rq member from each task_struct in the system:

```
  crash> list task_struct.tasks -s task_struct.se.on_rq -h ffff8800b66a0000
  ffff8800b66a0000
    se.on_rq = 1,
  ffff8800b66a0ad0
    se.on_rq = 0,
  ffff8800b66a15a0
    se.on_rq = 0,
  ffff8800b66a2070
    se.on_rq = 0,
  ffff8800b66a2b40
    se.on_rq = 0,
  ffff8800b67315a0
    se.on_rq = 0,
  ffff8800b6732b40
    se.on_rq = 0,
  ...
```

The task_struct.tasks example above requires that the -h option be given the address of a task_struct. Alternatively, the -l option can be given the address of a list_head or similar linkage structure whose first member points to the next linkage structure. Again using the task_struct.tasks embedded list_head, dump the "comm" member of all tasks by using -l in conjunction with -s option:

```
  crash> task -R tasks.next
  PID: 7044   TASK: ffff88005ac10000  CPU: 2   COMMAND: "crash"
    tasks.next = 0xffff880109b8e3d0,
  crash> list 0xffff880109b8e3d0 -l task_struct.tasks -s task_struct.comm
  ffff880109b8e3d0
    comm = "kworker/1:2"
  ffff880109b8be00
    comm = "bash"
  ffff88019d26c590
    comm = "cscope"
  ffff880109b8b670
    comm = "kworker/0:1"
  ffff880109b8cd20
    comm = "kworker/1:0"
  ffff88005ac15c40
    comm = "vi"
  ffff88005ac11fc0
    comm = "sleep"
  ffffffff81c135c0
    comm = "swapper/0"
  ffff880212828180
    comm = "systemd"
  ...
  ffff8801288d1830
    comm = "chrome"
  ffff8801534dd4b0
    comm = "kworker/0:0"
  ffff8801534d8180
    comm = "kworker/1:1"
  ffff88010902b670
    comm = "kworker/2:2"
  ffff880109b8a750
    comm = "sudo"
  ffff88005ac10180
    comm = "crash"
```

To display a liked list whose head node and other nodes are embedded within either same or different data structures resulting in different offsets for head node and other nodes, e.g. dentry.d_subdirs and dentry.d_child, the -O option can be used:

```
  crash> list -o dentry.d_child -s dentry.d_name.name -O dentry.d_subdirs -h
  ffff9c585b81a180
  ffff9c585b9cb140
    d_name.name = 0xffff9c585b9cb178 ccc.txt
  ffff9c585b9cb980
```

```
        d_name.name = 0xffff9c585b9cb9b8 bbb.txt
      ffff9c585b9cb740
        d_name.name = 0xffff9c585b9cb778 aaa.txt

  The dentry.d_subdirs example above is equal to the following sequence:

    crash> struct -o dentry.d_subdirs ffff9c585b81a180
    struct dentry {
      [ffff9c585b81a220] struct list_head d_subdirs;
    }
    crash> list -o dentry.d_child -s dentry.d_name.name -H ffff9c585b81a220


NAME
  rd - read memory

SYNOPSIS
  rd [-adDsSupxmfNR][-8|-16|-32|-64][-o offs][-e addr][-r file][address|symbol]
     [count]

DESCRIPTION
  This command displays the contents of memory, with the output formatted
  in several different manners.  The starting address may be entered either
  symbolically or by address.  The default output size is the size of a long
  data type, and the default output format is hexadecimal.  When hexadecimal
  output is used, the output will be accompanied by an ASCII translation.

       -p  address argument is a physical address.
       -u  address argument is a user virtual address; only required on
           processors with common user and kernel virtual address spaces.
       -m  address argument is a xen host machine address.
       -f  address argument is a dumpfile offset.
       -d  display output in signed decimal format (default is hexadecimal).
       -D  display output in unsigned decimal format (default is hexadecimal).
       -s  displays output symbolically when appropriate.
    -S[S]  displays output symbolically when appropriate; if the memory
           contents reference a slab cache object, the name of the slab cache
           will be displayed in brackets.  If -S is entered twice, and the
           memory contents reference a slab cache object, both the memory
           contents and the name of the slab cache will be displayed in
           brackets.
       -x  do not display ASCII translation at end of each line.
       -8  display output in 8-bit values.
      -16  display output in 16-bit values.
      -32  display output in 32-bit values (default on 32-bit machines).
      -64  display output in 64-bit values (default on 64-bit machines).
       -a  display output in ASCII characters if the memory contains printable
           ASCII characters; if no count argument is entered, stop at the first
           non-printable character.
       -N  display output in network byte order (only valid for 16- and 32-bit
           values)
       -R  display memory in reverse order; memory will be displayed up to and
           including the address argument, requiring the count argument to be
           greater than 1 in order to display memory before the specified
           address.
  -o offs  offset the starting address by offs.
  -e addr  display memory until reaching specified ending hexadecimal address.
  -r file  dumps raw data to the specified output file; the number of bytes that
           are copied to the file must be specified either by a count argument
           or by the -e option.
  address  starting hexadecimal address:
             1  the default presumes a kernel virtual address.
             2. -p specifies a physical address.
             3. -u specifies a user virtual address, but is only necessary on
                processors with common user and kernel virtual address spaces.
   symbol  symbol of starting address to read.
    count  number of memory locations to display; if entered, it must be the
           last argument on the command line; if not entered, the count defaults
           to 1, or unlimited for -a; when used with the -r option, it is the
           number of bytes to be written to the file.

EXAMPLES
  Display the kernel's version string:
```

```
  crash> rd -a linux_banner
  c082a020:  Linux version 2.6.32-119.el6.i686 (mockbuild@hs20-bc2-4.buil
  c082a05c:  d.redhat.com) (gcc version 4.4.4 20100726 (Red Hat 4.4.4-13)
  c082a098:   (GCC) ) #1 SMP Tue Mar 1 18:16:57 EST 2011
```

Display the same block of memory, first without symbols, again with symbols, and then with symbols and slab cache references:

```
  crash> rd f6e31f70 28
  f6e31f70:  f6e31f6c f779c180 c04a4032 00a9dd40   l.....y.2@J.@...
  f6e31f80:  00000fff c0472da0 f6e31fa4 f779c180   .....-G.......y.
  f6e31f90:  fffffff7 00a9b70f f6e31000 c04731ee   .............1G.
  f6e31fa0:  f6e31fa4 00000000 00000000 00000000   ................
  f6e31fb0:  00000000 00a9dd40 c0404f17 00000000   ....@....O@.....
  f6e31fc0:  00a9dd40 00000fff 00a9dd40 00a9b70f   @.......@.......
  f6e31fd0:  bf9e2718 ffffffda c040007b 0000007b   .'......{.@.{...
  crash> rd -s f6e31f70 28
  f6e31f70:  f6e31f6c f779c180 kmsg_read 00a9dd40
  f6e31f80:  00000fff vfs_read+159 f6e31fa4 f779c180
  f6e31f90:  fffffff7 00a9b70f f6e31000 sys_read+60
  f6e31fa0:  f6e31fa4 00000000 00000000 00000000
  f6e31fb0:  00000000 00a9dd40 syscall_call+7 00000000
  f6e31fc0:  00a9dd40 00000fff 00a9dd40 00a9b70f
  f6e31fd0:  bf9e2718 ffffffda startup_32+123 0000007b
  crash> rd -S f6e31f70 28
  f6e31f70:  [size-4096] [filp]   kmsg_read 00a9dd40
  f6e31f80:  00000fff vfs_read+159 [size-4096] [filp]
  f6e31f90:  fffffff7 00a9b70f [size-4096] sys_read+60
  f6e31fa0:  [size-4096] 00000000 00000000 00000000
  f6e31fb0:  00000000 00a9dd40 syscall_call+7 00000000
  f6e31fc0:  00a9dd40 00000fff 00a9dd40 00a9b70f
  f6e31fd0:  bf9e2718 ffffffda startup_32+123 0000007b
  crash> rd -SS f6e31f70 28
  f6e31f70:  [f6e31f6c:size-4096] [f779c180:filp] kmsg_read 00a9dd40
  f6e31f80:  00000fff vfs_read+159 [f6e31fa4:size-4096] [f779c180:filp]
  f6e31f90:  fffffff7 00a9b70f [f6e31000:size-4096] sys_read+60
  f6e31fa0:  [f6e31fa4:size-4096] 00000000 00000000 00000000
  f6e31fb0:  00000000 00a9dd40 syscall_call+7 00000000
  f6e31fc0:  00a9dd40 00000fff 00a9dd40 00a9b70f
  f6e31fd0:  bf9e2718 ffffffda startup_32+123 0000007b
```

Read jiffies in hexadecimal and decimal format:

```
  crash> rd jiffies
  c0213ae0:  0008cc3a                             :...

  crash> rd -d jiffies
  c0213ae0:        577376
```

Access the same memory in different sizes:

```
  crash> rd -64 kernel_version
  c0226a6c:  35312d352e322e32                    2.2.5-15

  crash> rd -32 kernel_version 2
  c0226a6c:  2e322e32 35312d35                     2.2.5-15

  crash> rd -16 kernel_version 4
  c0226a6c:  2e32 2e32 2d35 3531                       2.2.5-15

  crash> rd -8 kernel_version 8
  c0226a6c:  32 2e 32 2e 35 2d 31 35                           2.2.5-15
```

Read the range of memory from c009bf2c to c009bf60:

```
  crash> rd c009bf2c -e c009bf60
  c009bf2c:  c009bf64 c01328c3 c009bf64 c0132838   d....(..d...8(..
  c009bf3c:  0000002a 00000004 c57d77e8 00000104   *........w}.....
  c009bf4c:  0000000b c009a000 7fffffff 00000000   ................
  c009bf5c:  00000000                              ....
```

```
NAME
  task - task_struct and thread_info contents

SYNOPSIS
  task [-R member[,member]] [-dx] [pid | taskp] ...

DESCRIPTION
  This command dumps a formatted display of the contents of a task's
  task_struct and thread_info structures.  Multiple task or PID numbers
  may be entered; if no arguments are entered, the task_struct and
  thread_info structures of the current context are displayed.  The -R option,
  which may also be invoked indirectly via "foreach task", pares the output
  down to one or more structure members.

        pid  a process PID.
      taskp  a hexadecimal task_struct pointer.
  -R member  a comma-separated list of one or more task_struct and/or
             thread_info structure members.  If any member contains an embedded
             structure, or is an array, the output may be restricted to the
             embedded structure or an array element by expressing the member
             argument as "member.member" or "member[index]"; embedded member
             specifications may extend beyond one level deep, by expressing the
             member argument as "member.member.member...".
         -x  override default output format with hexadecimal format.
         -d  override default output format with decimal format.

EXAMPLES
  Dump the task_struct and thread_info structures of the current context
  in hexadecimal format:

    crash> task -x
    PID: 3176   TASK: f2451550  CPU: 1   COMMAND: "memtest"
    struct task_struct {
      state = 0x0,
      stack = 0xf05b6000,
      usage = {
        counter = 0x2
      },
      flags = 0x402040,
      ptrace = 0x0,
      lock_depth = 0xffffffff,
      prio = 0x78,
      static_prio = 0x78,
      normal_prio = 0x78,
      rt_priority = 0x0,
    ...
      perf_event_ctxp = {0x0, 0x0},
      memcg_batch = {
        do_batch = 0x0,
        memcg = 0x0,
        bytes = 0x0,
        memsw_bytes = 0x0
      }
    }

    struct thread_info {
      task = 0xf2451550,
      exec_domain = 0xc0a60860,
      flags = 0x88,
      status = 0x0,
      cpu = 0x1,
      preempt_count = 0x4010000,
      addr_limit = {
        seg = 0xc0000000
      },
      restart_block = {
    ...

  Display the ngroups and groups task_struct members for PID 2958:

    crash> task -R ngroups,groups 2958
    PID: 2958   TASK: c6718000  CPU: 0   COMMAND: "bash"
      ngroups = 6,
```

```
        groups = {504, 8, 9, 1000, 1007, 1006, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

    Display the embedded sched_entity structure's on_rq member:

      crash> task -R se.on_rq
      PID: 6529   TASK: ffff880116538790  CPU: 1   COMMAND: "bash"
        se.on_rq = 1,

    Display the 3rd pid_link structure in the embedded pids[] array:

    crash> task -R pids[2]
    PID: 6529   TASK: ffff880116538790  CPU: 0   COMMAND: "bash"
      pids[2] =   {
      node = {
        next = 0xffff8801165391b0,
        pprev = 0xffff880209d011b0
      },
      pid = 0xffff8801f0876e00
    }

    NOTE: When this command is invoked directly (i.e., not from "foreach"), it
    is not necessary to include the "-R" before the task_struct/thread_info
    member name(s).


  NAME
    exit - exit this session

  SYNOPSIS
    exit

  DESCRIPTION
    Bail out of the current crash session.

  NOTE
    This command is equivalent to the "q" command.


  NAME
    log - dump system message buffer

  SYNOPSIS
    log [-Ttdmas]

  DESCRIPTION
    This command dumps the kernel log_buf contents in chronological order.  The
    command supports the older log_buf formats, which may or may not contain a
    timestamp inserted prior to each message, as well as the newer variable-length
    record format, where the timestamp is contained in each log entry's header.

      -T  Display the message text with human readable timestamp.
          (Be aware that the timestamp could be inaccurate!  The timestamp is
           from local_clock(), which is different from the elapsed wall time.)
      -t  Display the message text without the timestamp; only applicable to the
          variable-length record format.
      -d  Display the dictionary of key/value pair properties that are optionally
          appended to a message by the kernel's dev_printk() function; only
          applicable to the variable-length record format.
      -m  Display the message log level in brackets preceding each message.  For
          the variable-length record format, the level will be displayed in
          hexadecimal.  In older kernels, by default, the facility/flag bits
          will be stripped to only show the level, but if needed, can still be
          shown with 'set debug 1'.
      -a  Dump the audit logs remaining in kernel audit buffers that have not
          been copied out to the user-space audit daemon.
      -s  Dump the printk logs remaining in kernel safe per-CPU buffers that
          have not been flushed out to log_buf.


  EXAMPLES
    Dump the kernel message buffer:
```

```
  crash> log
  Linux version 2.2.5-15smp (root@mclinux1) (gcc version egcs-2.91.66 19990
  314/Linux (egcs-1.1.2 release)) #1 SMP Thu Aug 26 11:04:37 EDT 1999
  Intel MultiProcessor Specification v1.4
      Virtual Wire compatibility mode.
  OEM ID: DELL     Product ID: WS 410       APIC at: 0xFEE00000
  Processor #0 Pentium(tm) Pro APIC version 17
  Processor #1 Pentium(tm) Pro APIC version 17
  I/O APIC #2 Version 17 at 0xFEC00000.
  Processors: 2
  mapped APIC to ffffe000 (fee00000)
  mapped IOAPIC to ffffd000 (fec00000)
  Detected 447696347 Hz processor.
  Console: colour VGA+ 80x25
  Calibrating delay loop... 445.64 BogoMIPS
  ...
    8K byte-wide RAM 5:3 Rx:Tx split, autoselect/Autonegotiate interface.
    MII transceiver found at address 24, status 782d.
    Enabling bus-master transmits and whole-frame receives.
  Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
  nfsd_init: initialized fhcache, entries=256
  ...
```

Do the same thing, but also show the log level preceding each message:

```
  crash> log -m
  <4>Linux version 2.2.5-15smp (root@mclinux1) (gcc version egcs-2.91.66 19990
  314/Linux (egcs-1.1.2 release)) #1 SMP Thu Aug 26 11:04:37 EDT 1999
  <4>Intel MultiProcessor Specification v1.4
  <4>    Virtual Wire compatibility mode.
  <4>OEM ID: DELL     Product ID: WS 410       APIC at: 0xFEE00000
  <4>Processor #0 Pentium(tm) Pro APIC version 17
  <4>Processor #1 Pentium(tm) Pro APIC version 17
  <4>I/O APIC #2 Version 17 at 0xFEC00000.
  <4>Processors: 2
  <4>mapped APIC to ffffe000 (fee00000)
  <4>mapped IOAPIC to ffffd000 (fec00000)
  <4>Detected 447696347 Hz processor.
  <4>Console: colour VGA+ 80x25
  <4>Calibrating delay loop... 445.64 BogoMIPS
  ...
  <6>  8K byte-wide RAM 5:3 Rx:Tx split, autoselect/Autonegotiate interface.
  <6>  MII transceiver found at address 24, status 782d.
  <6>  Enabling bus-master transmits and whole-frame receives.
  <6>Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
  <7>nfsd_init: initialized fhcache, entries=256
  ...
```

On a system with the variable-length record format, and whose log_buf has been filled and wrapped around, display the log with timestamp data:

```
  crash> log
  [    0.467730] pci 0000:ff:02.0: [8086:2c10] type 00 class 0x060000
  [    0.467749] pci 0000:ff:02.1: [8086:2c11] type 00 class 0x060000
  [    0.467769] pci 0000:ff:02.4: [8086:2c14] type 00 class 0x060000
  [    0.467788] pci 0000:ff:02.5: [8086:2c15] type 00 class 0x060000
  [    0.467809] pci 0000:ff:03.0: [8086:2c18] type 00 class 0x060000
  [    0.467828] pci 0000:ff:03.1: [8086:2c19] type 00 class 0x060000
  ...
```

Display the same message text as above, without the timestamp data:

```
  crash> log -t
  pci 0000:ff:02.0: [8086:2c10] type 00 class 0x060000
  pci 0000:ff:02.1: [8086:2c11] type 00 class 0x060000
  pci 0000:ff:02.4: [8086:2c14] type 00 class 0x060000
  pci 0000:ff:02.5: [8086:2c15] type 00 class 0x060000
  pci 0000:ff:03.0: [8086:2c18] type 00 class 0x060000
  pci 0000:ff:03.1: [8086:2c19] type 00 class 0x060000
  ...
```

Display the same message text as above, with appended dictionary data:

```
  crash> log -td
  pci 0000:ff:02.0: [8086:2c10] type 00 class 0x060000
  SUBSYSTEM=pci
  DEVICE=+pci:0000:ff:02.0
  pci 0000:ff:02.1: [8086:2c11] type 00 class 0x060000
  SUBSYSTEM=pci
  DEVICE=+pci:0000:ff:02.1
  pci 0000:ff:02.4: [8086:2c14] type 00 class 0x060000
  SUBSYSTEM=pci
  DEVICE=+pci:0000:ff:02.4
  pci 0000:ff:02.5: [8086:2c15] type 00 class 0x060000
  SUBSYSTEM=pci
  DEVICE=+pci:0000:ff:02.5
  pci 0000:ff:03.0: [8086:2c18] type 00 class 0x060000
  SUBSYSTEM=pci
  DEVICE=+pci:0000:ff:03.0
  pci 0000:ff:03.1: [8086:2c19] type 00 class 0x060000
  SUBSYSTEM=pci
  DEVICE=+pci:0000:ff:03.1
  ...
```

Dump the kernel audit logs:

```
  crash> log -a
  type=1320 audit(1489384479.809:4342):
  type=1300 audit(1489384479.809:4343): arch=c000003e syscall=0 success=yes
  exit=0 a0=4 a1=7f84154a2000 a2=400 a3=22 items=0 ppid=2560 pid=2591 auid=0
  uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=ttyS0 ses=1
  comm="pidof" exe="/usr/sbin/killall5"
  subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
  type=1320 audit(1489384479.809:4343):
  type=1300 audit(1489384479.809:4344): arch=c000003e syscall=3 success=yes
  exit=0 a0=4 a1=1 a2=8 a3=0 items=0 ppid=2560 pid=2591 auid=0 uid=0 gid=0
  euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=ttyS0 ses=1 comm="pidof"
  exe="/usr/sbin/killall5"
  subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
  type=1320 audit(1489384479.809:4344):
  type=1300 audit(1489384479.809:4345): arch=c000003e syscall=11
  success=yes exit=0 a0=7f84154a2000 a1=1000 a2=0 a3=0 items=0 ppid=2560
  pid=2591 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
  tty=ttyS0 ses=1 comm="pidof" exe="/usr/sbin/killall5"
  subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
  type=1320 audit(1489384479.809:4345):
  type=1300 audit(1489384479.809:4346): arch=c000003e syscall=2 success=yes
  exit=4 a0=7ffcfd20f5a0 a1=0 a2=1b6 a3=24 items=1 ppid=2560 pid=2591 auid=0
  uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=ttyS0 ses=1
  comm="pidof" exe="/usr/sbin/killall5"
  subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
  type=1307 audit(1489384479.809:4346):  cwd="/proc"
  ...
```

Display the message text with human readable timestamp:

```
  crash> log -T
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff]
  usable
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff]
  reserved
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff]
  reserved
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x0000000000100000-0x00000000dffeffff]
  usable
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x00000000dfff0000-0x00000000dfffffff]
  ACPI data
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff]
  reserved
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff]
  reserved
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff]
  reserved
  [Sat Apr  4 07:41:09 2020] BIOS-e820: [mem 0x0000000100000000-0x000000011fffffff]
  usable
  [Sat Apr  4 07:41:09 2020] NX (Execute Disable) protection: active
```

```
      [Sat Apr  4 07:41:09 2020] SMBIOS 2.5 present.
      [Sat Apr  4 07:41:09 2020] DMI: innotek GmbH VirtualBox/VirtualBox, BIOS
      VirtualBox 12/01/2006
      [Sat Apr  4 07:41:09 2020] Hypervisor detected: KVM
      [Sat Apr  4 07:41:09 2020] kvm-clock: Using msrs 4b564d01 and 4b564d00
      [Sat Apr  4 07:41:09 2020] kvm-clock: cpu 0, msr 6de01001, primary cpu clock
      [Sat Apr  4 07:41:09 2020] kvm-clock: using sched offset of 11838753697 cycles
      [Sat Apr  4 07:41:09 2020] clocksource: kvm-clock: mask: 0xffffffffffffffff
      max_cycles: 0x1cd42e4dffb, max_idle_ns: 881590591483 ns
      [Sat Apr  4 07:41:09 2020] e820: update [mem 0x00000000-0x00000fff] usable ==>
      reserved
      [Sat Apr  4 07:41:09 2020] e820: remove [mem 0x000a0000-0x000fffff] usable
      [Sat Apr  4 07:41:09 2020] last_pfn = 0x120000 max_arch_pfn = 0x400000000
      [Sat Apr  4 07:41:09 2020] MTRR default type: uncachable
      [Sat Apr  4 07:41:09 2020] MTRR variable ranges disabled:
      ...

    On a system which has printk_safe_seq_buf buffer,
    display its unflushed log with buffer name:

      crash> log
      ...
      [nmi_print_seq] Uhhuh. NMI received for unknown reason 30 on CPU 0.
      [nmi_print_seq] Do you have a strange power saving mode enabled?
      [nmi_print_seq] Dazed and confused, but trying to continue

    Dump the printk safe buffers:

      crash> log -s
      PRINTK_SAFE_SEQ_BUF: nmi_print_seq
      CPU: 0  ADDR: ffff8ca4fbc19ce0 LEN: 150  MESSAGE_LOST: 0
        Uhhuh. NMI received for unknown reason 20 on CPU 0.
        Do you have a strange power saving mode enabled?
        Dazed and confused, but trying to continue
      ...
      PRINTK_SAFE_SEQ_BUF: safe_print_seq
      CPU: 0  ADDR: ffff8ca4fbc1ad00 LEN: 0  MESSAGE_LOST: 0
        (empty)
      ...


NAME
  repeat - repeat a command

SYNOPSIS
  repeat [-seconds] command

DESCRIPTION
  This command repeats a command indefinitely, optionally delaying a given
  number of seconds between each command execution.

    -seconds   The number of seconds to delay between command executions.
               This option must precede the command name to be executed.

  Command execution may be stopped with CTRL-C, or if scrolling is in effect,
  by entering "q".  This command is meant for use on a live system; it is
  hard to conceive of a reason to use it when debugging a crash dump.

EXAMPLES
  Display the value of jiffies once per second:

    crash> repeat -1 p jiffies
    jiffies = $1 = 155551079
    jiffies = $2 = 155551180
    jiffies = $3 = 155551281
    jiffies = $4 = 155551382
    jiffies = $5 = 155551483
    jiffies = $6 = 155551584
    jiffies = $7 = 155551685
    jiffies = $8 = 155551786
    jiffies = $9 = 155551887
    jiffies = $10 = 155551988
    jiffies = $11 = 155552089
```

```
      jiffies = $12 = 155552190
      jiffies = $13 = 155552291
      jiffies = $14 = 155552392
      jiffies = $15 = 155552493
      jiffies = $16 = 155552594
      jiffies = $17 = 155552695
      jiffies = $18 = 155552796
      ...


NAME
  timer - timer queue data

SYNOPSIS
  timer [-r][-C cpu]

DESCRIPTION
  This command displays the timer queue entries, both old- and new-style,
  in chronological order.  In the case of the old-style timers, the
  timer_table array index is shown; in the case of the new-style timers,
  the timer_list address is shown.  On later kernels, the timer data is
  per-cpu.

    -r  Display hrtimer timer queue entries, both old- and new-style, in
        chronological order.  In the case of the old-style hrtimers, the
        expiration time is a single value; in the new-style hrtimers, the
        expiration time is a range.
 -C cpu Restrict the output to one or more CPUs, where multiple cpu[s] can
        be specified, for example, as "1,3,5", "1-3", or "1,3,5-7,10".

EXAMPLES
 Display the timer queue on an SMP system:

    crash> timer
    JIFFIES
    4296291038
    ...
    TIMER_BASES[1][BASE_STD]: ffff9801aba5aa00
      EXPIRES        TTE         TIMER_LIST     FUNCTION
      4296282997    -8041  ffff9801aba55ce0  ffffffff83a3bda0  <mce_timer_fn>
      4296283104    -7934  ffff97fd84bd35e0  ffffffff83ac6b70  <delayed_work_timer_fn>
      4296291061       23  ffffa6b283967de0  ffffffff83b29880  <process_timeout>
      4296291112       74  ffff9800c9b62ad8  ffffffff83e6b550  <cursor_timer_handler>
      4296291345      307  ffff980186d5ef88  ffffffff84146b80  <tcp_keepalive_timer>
      4296291484      446  ffff9801a7c54740  ffffffff84147f50  <tcp_write_timer>
      4296291997      959  fffffffffc073f880  ffffffff83ac6b70  <delayed_work_timer_fn>
      4296296213     5175  ffffa6b28339be18  ffffffff83b29880  <process_timeout>
      4296304383    13345  ffff980194ca72a8  ffffffff8412e4e0  <tw_timer_handler>
      4296305724    14686  ffff980194ca6918  ffffffff8412e4e0  <tw_timer_handler>
      4296306036    14998  ffff980194ca6d58  ffffffff8412e4e0  <tw_timer_handler>
      4296306883    15845  ffff980194ca7e58  ffffffff8412e4e0  <tw_timer_handler>
      4296307588    16550  ffff9801aaa27e58  ffffffff8412e4e0  <tw_timer_handler>
      4296307625    16587  ffff980194ca6a28  ffffffff8412e4e0  <tw_timer_handler>
      4296313542    22504  ffff980194ca7c38  ffffffff8412e4e0  <tw_timer_handler>
      4296317680    26642  ffff9800c9149c58  ffffffff840da870  <neigh_timer_handler>
      4296317744    26706  ffff9801a5354468  ffffffff83ac6b70  <delayed_work_timer_fn>
      4296343322    52284  ffff980194ca63c8  ffffffff8412e4e0  <tw_timer_handler>
      4296343581    52543  ffff980194ca7088  ffffffff8412e4e0  <tw_timer_handler>
      4296343597    52559  ffff9801aaa274c8  ffffffff8412e4e0  <tw_timer_handler>
      4296714205   423167  fffffffff84caf3c0  ffffffff83ac6b70  <delayed_work_timer_fn>
    TIMER_BASES[1][BASE_DEF]: ffff9801aba5bc80
      EXPIRES        TTE         TIMER_LIST     FUNCTION
      4296291264      226  ffffffff855eb238  ffffffff83c08fb0  <writeout_period>
      4296319997    28959  fffffffffc06ede40  ffffffff83ac6b70  <delayed_work_timer_fn>
      4296506084   215046  ffff9801aba629c8  ffffffff83ac5ea0  <idle_worker_timeout>
    ...

  Display a new-style hrtimer queue:

    crash> timer -r
    ...
    CPU: 2  HRTIMER_CPU_BASE: ffff9801aba9cf00
      CLOCK: 0  HRTIMER_CLOCK_BASE: ffff9801aba9cf40  [ktime_get]
```

```
          CURRENT
        1623742000000
         SOFTEXPIRES      EXPIRES         TTE         HRTIMER           FUNCTION
        1623741000000  1623741000000    -1000000  ffff9801aba9d540  ffffffff83b3c8e0
        <tick_sched_timer>
        1624024000000  1624024000000   282000000  ffff9801aba9d720  ffffffff83b7e7a0
        <watchdog_timer_fn>
        1626000939806  1626010929804  2268929804  ffffa6b28399fa40  ffffffff83b2c1e0
        <hrtimer_wakeup>
        1627576915615  1627576915615  3834915615  ffff9801a5727978  ffffffff83b365c0
        <posix_timer_fn>
        1627637194488  1627647194487  3905194487  ffffa6b283977db0  ffffffff83b2c1e0
        <hrtimer_wakeup>
        1629937423000  1629937423000  6195423000  ffff9801a9af2900  ffffffff83cf3d30
        <timerfd_tmrproc>

        CLOCK: 1  HRTIMER_CLOCK_BASE: ffff9801aba9cf80  [ktime_get_real]
              CURRENT
        1558362388334558243
            SOFTEXPIRES            EXPIRES             TTE           HRTIMER
            FUNCTION
        1558362389331238000  1558362389331288000      996729757  ffffa6b28574bcf0
        ffffffff83b2c1e0  <hrtimer_wakeup>
        1558364372000000000  1558364372000000000  1983665441757  ffff9801a3513278
        ffffffff83b365c0  <posix_timer_fn>

        CLOCK: 2  HRTIMER_CLOCK_BASE: ffff9801aba9cfc0  [ktime_get_boottime]
        (empty)
      ...
```

```
NAME
  extend - extend the crash command set

SYNOPSIS
  extend [shared-object ...] | [-u [shared-object ...]] | -s

DESCRIPTION
  This command dynamically loads or unloads crash extension shared object
  libraries:

    shared-object     load the specified shared object file; more than one
                      one object file may be entered.
    -u shared-object  unload the specified shared object file; if no file
                      arguments are specified, unload all objects.
    -s                show all available shared object files.

  If the shared-object filename is not expressed with a fully-qualified
  pathname, the following directories will be searched in the order shown,
  and the first instance of the file that is found will be selected:

     1. the current working directory
     2. the directory specified in the CRASH_EXTENSIONS environment variable
     3. /usr/lib64/crash/extensions (64-bit architectures)
     4. /usr/lib/crash/extensions
     5. the ./extensions subdirectory of the current directory

  If no arguments are entered, the current set of shared object files and
  a list of their commands will be displayed.  The registered commands
  contained in each shared object file will appear automatically in the
  "help" command screen.

  An example of a shared object prototype file, and how to compile it
  into a shared object, is appended below.

EXAMPLES
  Load two shared object files:

    crash> extend extlib1.so extlib2.so
    ./extlib1.so: shared object loaded
    ./extlib2.so: shared object loaded
```

```
  Display the current set of shared object files and their commands:

    crash> extend
    SHARED OBJECT  COMMANDS
    ./extlib1.so   echo util bin
    ./extlib2.so   smp show

  Unload one of the shared object files:

    crash> extend -u extlib1.so
    ./extlib1.so: shared object unloaded

  Unload all currently-loaded object files:

    crash> extend -u
    ./extlib2.so: shared object unloaded

CREATING A SHARED OBJECT
  The extend command loads shared object files using dlopen(3), which in
  turn calls the shared object's constructor function.  The shared object's
  constructor function should register its command set by calling
  register_extension(), passing it a pointer to an array of one or more
  structures of the following type:

    struct command_table_entry {
            char *name;
            cmd_func_t func;
            char **help_data,
            ulong flags;
    };

  Each command_table_entry structure contains the ASCII name of a command,
  the command's function address, a pointer to an array of help data strings,
  and a flags field.  The help_data field is optional; if it is non-NULL, it
  should point to an array of character strings used by the "help"
  command, and during command failures.  The flags field currently has two
  available bit settings, REFRESH_TASK_TABLE, which should be set if it is
  preferable to reload the current set of running processes just prior to
  executing the command (on a live system) and MINIMAL, which should be
  set if the command should be available in minimal mode.  Terminate the array
  of command_table_entry structures with an entry with a NULL command name.

  Below is an example shared object file consisting of just one command,
  called "echo", which simply echoes back all arguments passed to it.
  Note the comments contained within it for further details.  Cut and paste
  the following output into a file, and call it, for example, "echo.c".
  Then compiled in either of two manners.  Either manually like so:

 gcc -shared -rdynamic -o echo.so echo.c -fPIC -D<machine-type> $(TARGET_CFLAGS)

  where <machine-type> must be one of the MACHINE_TYPE #define's in defs.h,
  and where $(TARGET_CFLAGS) is the same as it is declared in the top-level
  Makefile after a build is completed.  Or alternatively, the "echo.c" file
  can be copied into the "extensions" subdirectory, and compiled automatically
  like so:

  make extensions

  The echo.so file may be dynamically linked into crash during runtime, or
  during initialization by putting "extend echo.so" into a .crashrc file
  located in the current directory, or in the user's $HOME directory.

--------------------------------- cut here ---------------------------------

#include "defs.h"      /* From the crash source top-level directory */

void echo_init(void);    /* constructor function */
void echo_fini(void);    /* destructor function (optional) */

void cmd_echo(void);     /* Declare the commands and their help data. */
char *help_echo[];

static struct command_table_entry command_table[] = {
```

```
7565            { "echo", cmd_echo, help_echo, 0},          /* One or more commands, */
7566            { NULL },                                     /* terminated by NULL, */
7567    };
7568
7569
7570    void __attribute__((constructor))
7571    echo_init(void) /* Register the command set. */
7572    {
7573            register_extension(command_table);
7574    }
7575
7576    /*
7577     *  This function is called if the shared object is unloaded.
7578     *  If desired, perform any cleanups here.
7579     */
7580    void __attribute__((destructor))
7581    echo_fini(void) { }
7582
7583
7584    /*
7585     *  Arguments are passed to the command functions in the global args[argcnt]
7586     *  array.  See getopt(3) for info on dash arguments.  Check out defs.h and
7587     *  other crash commands for usage of the myriad of utility routines available
7588     *  to accomplish what your task.
7589     */
7590    void
7591    cmd_echo(void)
7592    {
7593            int c;
7594
7595            while ((c = getopt(argcnt, args, "")) != EOF) {
7596                    switch(c)
7597                    {
7598                    default:
7599                            argerrs++;
7600                            break;
7601                    }
7602            }
7603
7604            if (argerrs)
7605                    cmd_usage(pc->curcmd, SYNOPSIS);
7606
7607            while (args[optind])
7608                    fprintf(fp, "%s ", args[optind++]);
7609
7610            fprintf(fp, "\n");
7611    }
7612
7613    /*
7614     *  The optional help data is simply an array of strings in a defined format.
7615     *  For example, the "help echo" command will use the help_echo[] string
7616     *  array below to create a help page that looks like this:
7617     *
7618     *    NAME
7619     *      echo - echoes back its arguments
7620     *
7621     *    SYNOPSIS
7622     *      echo arg ...
7623     *
7624     *    DESCRIPTION
7625     *      This command simply echoes back its arguments.
7626     *
7627     *    EXAMPLE
7628     *      Echo back all command arguments:
7629     *
7630     *        crash> echo hello, world
7631     *        hello, world
7632     *
7633     */
7634
7635    char *help_echo[] = {
7636            "echo",                         /* command name */
7637            "echoes back its arguments",    /* short description */
```

```
        "arg ...",                      /* argument synopsis, or " " if none */

        "  This command simply echoes back its arguments.",
        "\nEXAMPLE",
        "  Echo back all command arguments:\n",
        "    crash> echo hello, world",
        "    hello, world",
        NULL
};



NAME
  mach - machine specific data

SYNOPSIS
  mach [-m | -c -[xd] | -o]

DESCRIPTION
  This command displays data specific to a machine type.

    -m  Display the physical memory map (x86, x86_64 and ia64 only).
    -c  Display each cpu's cpuinfo structure (x86, x86_64 and ia64 only).
        Display each cpu's x8664_pda structure (x86_64 only),
        Display the hwrpb_struct, and each cpu's percpu_struct (alpha only).
    -x  override default output format with hexadecimal format.
    -d  override default output format with decimal format.
    -o  Display the OPAL console log (ppc64 only).

EXAMPLES
    crash> mach
           MACHINE TYPE: i686
            MEMORY SIZE: 512 MB
                   CPUS: 2
             HYPERVISOR: KVM
        PROCESSOR SPEED: 1993 Mhz
                     HZ: 100
              PAGE SIZE: 4096
    KERNEL VIRTUAL BASE: c0000000
    KERNEL VMALLOC BASE: e0800000
      KERNEL STACK SIZE: 8192

  Display the system physical memory map:

    crash> mach -m
          PHYSICAL ADDRESS RANGE         TYPE
    0000000000000000 - 00000000000a0000  E820_RAM
    00000000000f0000 - 0000000000100000  E820_RESERVED
    0000000000100000 - 000000001ff75000  E820_RAM
    000000001ff75000 - 000000001ff77000  E820_NVS
    000000001ff77000 - 000000001ff98000  E820_ACPI
    000000001ff98000 - 0000000020000000  E820_RESERVED
    00000000fec00000 - 00000000fec90000  E820_RESERVED
    00000000fee00000 - 00000000fee10000  E820_RESERVED
    00000000ffb00000 - 0000000100000000  E820_RESERVED

  Display the OPAL console log:

    crash> mach -o
    [   65.219056911,5] SkiBoot skiboot-5.4.0-218-ge0225cc-df9a248 starting...
    [   65.219065872,5] initial console log level: memory 7, driver 5
    [   65.219068917,6] CPU: P8 generation processor(max 8 threads/core)
    [   65.219071681,7] CPU: Boot CPU PIR is 0x0060 PVR is 0x004d0200
    [   65.219074685,7] CPU: Initial max PIR set to 0x1fff
    [   65.219607955,5] FDT: Parsing fdt @0xff00000
    [  494.026291523,7] BT: seq 0x25 netfn 0x0a cmd 0x48: Message sent to host
    [  494.027636927,7] BT: seq 0x25 netfn 0x0a cmd 0x48: IPMI MSG done


NAME
  runq - run queue

SYNOPSIS
```

```
  runq [-t] [-T] [-m] [-g] [-c cpu(s)]

DESCRIPTION
  With no argument, this command displays the tasks on the run queues
  of each cpu.

     -t  Display the timestamp information of each cpu's runqueue, which is the
         rq.clock, rq.most_recent_timestamp or rq.timestamp_last_tick value,
         whichever applies; following each cpu timestamp is the last_run or
         timestamp value of the active task on that cpu, whichever applies,
         along with the task identification.
     -T  Display the time lag of each CPU relative to the most recent runqueue
         timestamp.
     -m  Display the amount of time that the active task on each cpu has been
         running, expressed in a format consisting of days, hours, minutes,
         seconds and milliseconds.
     -g  Display tasks hierarchically by task_group.  The task_group line shows
         the task_group address, the cfs_rq or rt_rq address, the task_group
         name (if any), and whether the task_group is throttled.
 -c cpu  restrict the output to the run queue data of one or more CPUs,
         which can be specified using the format "3", "1,8,9", "1-23",
         or "1,8,9-14".

EXAMPLES
 Display the tasks on an O(1) scheduler run queue:

    crash> runq
    CPU 0 RUNQUEUE: ffff880001cdb460
      CURRENT: PID: 2739   TASK: ffff8800320fa7e0  COMMAND: "bash"
      ACTIVE PRIO_ARRAY: ffff880001cdb4d8
         [115] PID: 2739   TASK: ffff8800320fa7e0  COMMAND: "bash"
               PID: 1776   TASK: ffff88003217d820  COMMAND: "syslogd"
      EXPIRED PRIO_ARRAY: ffff880001cdbdb8
         [no tasks queued]

    CPU 1 RUNQUEUE: ffff880001ce3460
      CURRENT: PID: 1779   TASK: ffff88003207a860  COMMAND: "klogd"
      ACTIVE PRIO_ARRAY: ffff880001ce34d8
         [115] PID: 1779   TASK: ffff88003207a860  COMMAND: "klogd"
      EXPIRED PRIO_ARRAY: ffff880001ce3db8
         [no tasks queued]

 Display the tasks on a CFS run queue:

    crash> runq
    CPU 0 RUNQUEUE: ffff8800090436c0
      CURRENT: PID: 588    TASK: ffff88007e4877a0  COMMAND: "udevd"
      RT PRIO_ARRAY: ffff8800090437c8
         [no tasks queued]
      CFS RB_ROOT: ffff880009043740
         [118] PID: 2110   TASK: ffff88007d470860  COMMAND: "check-cdrom.sh"
         [118] PID: 2109   TASK: ffff88007f1247a0  COMMAND: "check-cdrom.sh"
         [118] PID: 2114   TASK: ffff88007f20e080  COMMAND: "udevd"

    CPU 1 RUNQUEUE: ffff88000905b6c0
      CURRENT: PID: 2113   TASK: ffff88007e8ac140  COMMAND: "udevd"
      RT PRIO_ARRAY: ffff88000905b7c8
         [no tasks queued]
      CFS RB_ROOT: ffff88000905b740
         [118] PID: 2092   TASK: ffff88007d7a4760  COMMAND: "MAKEDEV"
         [118] PID: 1983   TASK: ffff88007e59f140  COMMAND: "udevd"
         [118] PID: 2064   TASK: ffff88007e40f7a0  COMMAND: "udevd"
         [115] PID: 2111   TASK: ffff88007e4278a0  COMMAND: "kthreadd"

 Display run queue timestamp data:

    crash> runq -t
    CPU 0: 2680990637359
           2680986653330  PID: 28228  TASK: ffff880037ca2ac0  COMMAND: "loop"
    CPU 1: 2680940618478
           2680940618478  PID: 28167  TASK: ffff880078130040  COMMAND: "bash"
    CPU 2: 2680990763425
           2680986785772  PID: 28227  TASK: ffff8800787780c0  COMMAND: "loop"
```

```
      CPU 3: 2680990954469
             2680986059540  PID: 28226  TASK: ffff880078778b00  COMMAND: "loop"

 Display the amount of time the active task on each cpu has been running:

    crash> runq -m
     CPU 0: [0 00:00:00.014]  PID: 5275  TASK: f5dbcaa0  COMMAND: "sh"
     CPU 1: [0 00:00:00.002]  PID: 5203  TASK: f5c7baa0  COMMAND: "cat"
     CPU 2: [0 00:00:00.014]  PID: 7971  TASK: f5c6c550  COMMAND: "khelper"
     CPU 3: [0 00:00:00.002]  PID: 0     TASK: f4ccd000  COMMAND: "swapper"

 Display tasks hierarchically by task_group:

  crash> runq -g
  CPU 0
    CURRENT: PID: 14734  TASK: ffff88010626f500  COMMAND: "sh"
    ROOT_TASK_GROUP: ffffffff81ed93e0  RT_RQ: ffff880028216808
       [  0] TASK_GROUP: ffff88022c6bbc00 RT_RQ: ffff880139fc9800 (THROTTLED)
            [  0] PID: 14750  TASK: ffff88013a4dd540  COMMAND: "rtloop99"
            [  1] PID: 14748  TASK: ffff88013bbca040  COMMAND: "rtloop98"
            [  1] TASK_GROUP: ffff88012b0fb400 RT_RQ: ffff880089029000
                  [  1] PID: 14752  TASK: ffff880088abf500  COMMAND: "rtloop98"
            [ 54] PID: 14749  TASK: ffff880037a4e080  COMMAND: "rtloop45"
            [ 98] PID: 14746  TASK: ffff88012678c080  COMMAND: "rtloop1"
    ROOT_TASK_GROUP: ffffffff81ed93e0  CFS_RQ: ffff8800282166e8
       [120] PID: 14740  TASK: ffff88013b1e6080  COMMAND: "sh"
       [120] PID: 14738  TASK: ffff88012678d540  COMMAND: "sh"
       [120] PID: 14734  TASK: ffff88010626f500  COMMAND: "sh" [CURRENT]
       TASK_GROUP: ffff884052bc9800 CFS_RQ: ffff8831e4a1b000 (THROTTLED)
          [120] PID: 14732  TASK: ffff88013bbcb500  COMMAND: "sh"
          [120] PID: 14728  TASK: ffff8800b3496080  COMMAND: "sh"
          [120] PID: 14730  TASK: ffff880037833540  COMMAND: "sh"
       TASK_GROUP: ffff884058f1d000 CFS_RQ: ffff88120a101600 (THROTTLED)
          [120] PID: 14726  TASK: ffff880138d42aa0  COMMAND: "sh"
  ...

 Display tasks hierarchically by task_group for cpu 3 only:

  crash> runq -g -c3
  CPU 3
    CURRENT: PID: 2948   TASK: ffff88022af2a100  COMMAND: "bash"
    INIT_TASK_GROUP: ffffffff81e1a780  RT_RQ: ffff880028216148
       [no tasks queued]
    INIT_TASK_GROUP: ffffffff81e1a780  CFS_RQ: ffff880028216028
       [120] PID: 2948   TASK: ffff88022af2a100  COMMAND: "bash" [CURRENT]
       TASK_GROUP: ffff88012b880800  CFS_RQ: ffff88012c5d1000  <libvirt>
          TASK_GROUP: ffff88012c078000  CFS_RQ: ffff88012c663e00  <qemu>
             TASK_GROUP: ffff88022c7f4c00  CFS_RQ: ffff88012bb56000  <guest2>
                TASK_GROUP: ffff88022b621400  CFS_RQ: ffff88012b012000  <vcpu0>
                   [120] PID: 3248   TASK: ffff88012a9d4100  COMMAND: "qemu-kvm"


NAME
  trace - show or dump the tracing info

SYNOPSIS
  trace [ <show [-c <cpulist>] [-f [no]<flagname>]> | <dump [-sm] <dest-dir>> ] |
  <dump -t <trace.dat> ]

DESCRIPTION
trace
    shows the current tracer and other informations.

trace show
    shows all events with readability text(sorted by timestamp)

trace report
    the same as "trace show"

trace dump [-sm] <dest-dir>
    dump ring_buffers to dest-dir. Then you can parse it
    by other tracing tools. The dirs and files are generated
    the same as debugfs/tracing.
```

```
      -m: also dump metadata of ftrace.
      -s: also dump symbols of the kernel.
  trace dump -t [output-file-name]
     dump ring_buffers and all meta data to a file that can
     be parsed by trace-cmd. Default output file name is "trace.dat".
```