NAME

ddb — kernel debugger

DESCRIPTION

The **ddb** debugger provides a means for debugging the kernel, and analysing the kernel after a system crash ("panic"), with a gdb(1)-like syntax.

ddb is invoked upon a kernel panic when the sysctl(8) *ddb.panic* is set to 1. It may be invoked from the console when the sysctl *ddb.console* is set to 1, using any of the following methods:

— Using the key sequence `Ctrl-Alt-Esc`.

— Sending a `BREAK` when using a serial console.

— Writing to the sysctl *ddb.trigger*.

— For i386 and amd64 architectures, using the key sequence `Ctrl-Alt-Delete` when the sysctl *machdep.kbdreset* is set to 2.

ddb prompts for commands on the console with:

```
ddb>
```

The general syntax of a **ddb** command is:

command [**/***modifiers*] [*address*][**,***count*]

To save typing, **ddb** makes use of a context inferred from previous commands. In this context, the current location is called *dot*. The **examine**, **search**, **show struct**, and **write** commands update *dot* to be that of the last address examined or the last location modified, and have intuitive effects on *next* and *prev*. All the other commands do not change *dot*, and set *next* to be the same. (See VARIABLES.)

An expression can be used in place of *address* (see EXPRESSIONS). Omitting *address* in a command uses the last value of *dot*. A missing *count* is taken to be 1 for printing commands or infinity for stack traces. Entering a blank line causes the last command to be repeated using *next* in place of *address*, a *count* of 1, and no modifiers.

ddb has a feature like more(1) for the output. If the number of lines output in response to one command exceeds the number set in the *$lines* variable, it displays the message '`--db_more--`' and waits for a response.

The valid responses are:

⟨**space**⟩
: One more page.

⟨**return**⟩
: One more line.

q
: Abort the current command, and return to the command input mode.

The following command line editing keys are provided:

^b back one character
^f forward one character
^a beginning of line
^e end of line
^w erase word back
^h | ⟨**del**⟩
: erase previous character

^d erase next character
^k delete to end of line
^u delete line
^p previous in command history
^n next in command history
^r redraw line
^t exchange the two characters to the left of the cursor

COMMANDS

The following commands may be typed at the '`ddb>`' prompt. Some commands consist of more than one word, and if only the first word or words are entered, the possible alternatives to complete the command are displayed and no other action is performed.

help
: List the available commands.

[**e**]**x**[**amine**] [**/bhlqaAxzodurcsmil**] [*addr*][**,***count*]
: Display the contents at address *addr* according to the formats in the modifier. If no format is specified, the last formats specified for this command are used.

  The format characters are:

  /b look at by bytes (8 bits)
  /h look at by half words (16 bits)
  /l look at by long words (32 bits) (default)
  /q look at by long longs (64 bits) (only available on 64-bit platforms)
  /a print the location being displayed
  /A print the location with a line number if possible
  /x display in unsigned hex

/z display in signed hex
/o display in unsigned octal
/d display in signed decimal
/u display in unsigned decimal
/r display in current radix, signed
/c display low 8 bits as a character. Non-printing characters are displayed as an octal escape code (e.g., '\000').
/s display the null-terminated string at the location. Non-printing characters are displayed as octal escapes.
/m display in unsigned hex with character dump at the end of each line. The location is also displayed in hex at the beginning of each line.
/i display as an instruction
/I display as an alternate format instruction depending on the machine:

alpha
: Print affected register contents for every instruction.

amd64,
i386
: Do not skip padding to the next long word boundary for unconditional jumps.

m88k
: Decode instructions for the opposite CPU model (e.g. m88110 when running on an m88100 processor).

The value of *next* is set to the *addr* plus the size of the data examined.

p[**rint**] [**/axzodurc**] [*addr*]
Print *addr* according to the modifier character. The valid modifiers are a subset of those from the **examine** command, and act as described there. If no modifier is specified, the last one specified in a previous use of **print** is used.

For example,

```
print/x $eax
```

will print something like this:

```
xxxxxx
```

pp[**rint**] [*addr*]
Pretty-print *addr* using CTF debug symbols included in the kernel binary image. The CTF section is normally added by running ctfstrip(1) as part of building a new kernel.

w[**rite**] [**/bhl**] [*addr*] *expr* [*expr ...*]
Write the value of each *expr* expression at succeeding locations start at *addr*. The write unit size can be specified using one of the modifiers:

/b byte (8 bits)
/h half word (16 bits)
/l long word (32 bits) (default)

The value of *next* is set to *addr* plus the size of values written.

Warning: since there is no delimiter between expressions, the command may not parse as you expect. It is best to enclose each expression in parentheses.

set **$***name* [**=**] *expr*
Set the named variable or register with the value of *expr*. For valid variable names, see VARIABLES.

boot *how*
Reboot the machine depending on *how*:

boot sync
: Sync disks and reboot.

boot crash
: Dump core and reboot.

boot dump
: Sync disks, dump core and reboot.

boot halt
: Just halt.

boot reboot
: Just reboot.

boot poweroff
: Power down the machine whenever possible; if it fails, just halt.

break [*addr*][**,***count*]
Set a break point at *addr*. If *count* is supplied, **ddb** allows the breakpoint to be silently hit (*count* - 1) times before stopping at the break point.

d[**elete**] [*addr*]
Delete the break point set with the **break** command.

s[**tep**] [**/p**] [**,***count*]
Single step *count* times. If the **/p** modifier is specified, print each instruction at each step. Otherwise, only print the last instruction.

Warning: depending on machine type, it may not be possible to single-step through some low-level code paths. On machines with software-emulated single-stepping (e.g., alpha), stepping through code executed by interrupt handlers will probably do the wrong thing.

call *name***(***expr* [**,** *expr ...*] **)**
Call the function named by *name* with the argument(s) listed in parentheses. Parentheses may be omitted if the function takes no arguments. The number of arguments is currently limited to 10.

c[**ontinue**] [**/c**]
Continue execution until a breakpoint or watchpoint. If the **/c** modifier is given, instructions are counted while executing.

Warning: when counting with **/c**, **ddb** is really silently single-stepping. This means that single-stepping on low-level code may cause strange behavior.

watch *addr* [**,***size*]
Set a watchpoint for the region starting at *addr*. Execution stops and control returns to **ddb** when an attempt is made to modify a watched

region. The *size* argument defaults to 4.

If you specify a wrong space address, the request is rejected with an error message.

Warning: attempts to watch wired kernel memory may cause an unrecoverable error on some systems (e.g., i386).

dwatch *addr*
Delete the watchpoint at address *addr* that was previously set with a **watch** command.

hangman [**/s**[**0-9**]]
This is a tiny and handy tool for random kernel hangs analysis, of which its depth is controlled by the optional argument of the default value of five. It uses some sophisticated heuristics to spot the global symbol that caused the hang. Since the discovering algorithm is a probabilistic one, you may spend substantial time to figure the exact symbol name. This smart thing requires a little of your attention, the input it accepts is mostly of the same format as that of the famous hangman(6) game, to which it, apparently, is obliged by the name. Hint: the nm(1) utility might help.

until [**/p**]
Stop at the next "call" or "return" instruction. If the **/p** modifier is specified, **ddb** prints the call nesting depth and the cumulative instruction count at each call or return. Otherwise, it stays silent until the matching return is hit.

match [**/p**]
Stop at the next matching return instruction. If the **/p** modifier is specified, **ddb** prints the call nesting depth and the cumulative instruction count at each call or return. Otherwise, it remains mostly quiet.

next [**/p**]
The **next** command is a synonym for **match**.

kill *pid*
Send an uncatchable `SIGABRT` signal to the process specified by the *pid* argument.

trace [**/tu**] [*frameaddr*][**,***count*]
Show the stack trace. The **/t** modifier interprets the *frameaddr* argument as the TID of a process and shows the stack trace of that process. *frameaddr* is subject to the radix; use the 0t prefix to enter a decimal TID. The **/t** modifier is not supported on all platforms. The **/u** modifier shows the stack trace of user space; if omitted, the kernel stack is traced instead. The *count* argument is the limit on the number of frames to be followed. If *count* is omitted, all frames are printed.

Warning: user space stack trace is valid only if the machine dependent code supports it.

search [**/bhl**] [*addr*] *value* [*mask*] [**,***count*]
Search memory for a value beginning at *addr*. This command might fail in interesting ways if it doesn't find the searched-for value. This is because **ddb** doesn't always recover from touching bad memory. The optional *count* argument limits the search. The modifiers are the same as those of the **write** command.

The *next* address is set to the address where *value* is found, or just after where the search area finishes.

reboot
Shortcut for **boot reboot**

show *what*
Displays various things, depending on *what*:

show bcstats
Prints the buffer cache statistics.

show breaks
Prints a list of all breakpoints that have been set with the **break** command.

show buf [**/f**] *addr*
Prints the `struct buf` at *addr*. If the **/f** modifier is specified output will also include `softdep` printout, if those are available.

show extents
Prints a detailed list of all extents.

show locks [*addr*]
Prints the list of locks held by a thread. If an optional address is not specified, `curproc` is assumed. The **option WITNESS** is required for this command to be available.

show malloc [*addr*]
Prints malloc debugging information if available. If an optional address is specified, only information about that address is printed.

show map [**/f**] *addr*
Prints the `vm_map` at *addr*. If the **/f** modifier is specified the complete map is printed.

show mbuf *addr*
Prints the `struct mbuf` header at *addr*. Depending on the mbuf flags `struct pkthdr` and `struct m_ext` are printed as well.

show mount [**/f**] *addr*
Prints the `struct mount` at *addr*. If the **/f** modifier is specified prints out all `vnodes` (see also **show vnode**) and also all `bufs` (see also **show buf**) on all those vnodes.

show nfsnode [**/f**] *addr*
Prints the `struct nfsnode` at *addr*. If the **/f** modifier is specified prints out additional information as well.

show nfsreq [**/f**] *addr*
Prints the `struct nfsreq` at *addr*. If the **/f** modifier is specified prints out additional information as well.

show object [**/f**] *addr*
Prints the `vm_object` at *addr*. If the **/f** modifier is specified the complete object is printed.

show page [**/f**] *addr*
Prints the `vm_page` at *addr*. If the **/f** modifier is specified the complete page is printed.

show panic
Prints the panic string.

show pool [**/p**] *addr*
Prints the `pool` at *addr*. Valid modifiers:
/p Print the pagelist for this pool.

show proc [*addr*]
Prints the `struct proc` at *addr*. If an optional address is not specified `curproc` is assumed.

show registers [**/u**]
Display the register set. If the **/u** modifier is specified, it displays user registers (or the currently saved registers) instead of the kernel's. Note: The **/u** modifier is not supported on every machine, in which case incorrect information may be displayed.

show socket *addr*
Prints the `struct socket` at *addr*. If the socket is spliced, the `struct sosplice` associated with the socket is printed as well.

show struct *name* [addr]
Prints the content of the memory at *addr* as a struct *name*. Nested structures and bit fields are not printed. Character arrays are printed as bytes.

show uvmexp
Displays a selection of uvm counters and statistics.

show vnode [**/f**] *addr*
Prints the `struct vnode` at *addr*. If the **/f** modifier is specified prints out all `bufs` (see also **show buf**) currently attached to this `vnode`.

show watches
Displays all watchpoints set with the **watch** command.

show witness [**/b**]
Prints the current order list. If the **/b** modifier is specified, the list of found lock order violations is printed instead. The **option WITNESS** is required for this command to be available.

show all procs [**/anow**]
Display information on all processes.

- **/n** (Default) Show process information in a ps(1)-like format. Information printed includes process ID, thread ID, parent process ID, UID, process status, process flags, process wait channel message and process command name.
- **/a** Shows the kernel virtual addresses of each process' proc structure, u-area, and vmspace structure. The vmspace address is also the address of the process' `vm_map` structure and can be used in the **show map** command.
- **/o** Shows non-idle threads that were on CPU when ddb was entered. Information printed includes thread ID, process ID, UID, process flags, thread flags, current CPU, and command name.
- **/w** Shows each thread's ID, command, process group, wait channel address, and wait channel message.

show all bufs [**/f**]
Display information about all buffers in the system.

- **/f** For each buffer, print a more detailed output. See the **show buf** command for more information.

show all callout
Display the contents of the callout table.

show all pools [**/a**]
Display information about all system pools in a format similar to vmstat(8).

- **/a** Displays “interesting” address information.

show all locks
Prints the list of locks held by all threads in the system. The **option WITNESS** is required for this command to be available.

show all mounts [**/f**]
Display information on all mounted filesystems.

- **/f** For each filesystem, list all its struct vnode addresses. These addresses can be used in the **show vnode** command.

show all nfsnodes [**/f**]
Display information about all nfsnodes in the system.

- **/f** For each nfsnode, print a more detailed output. See the **show nfsnode** command for more information.

show all nfsreqs [**/f**]
Display information for all outstanding NFS requests.

- **/f** For each NFS requests, print a more detailed output. See the **show nfsreq** command for more information.

show all vnodes [**/f**]
Display information about all vnodes in the system.

- **/f** For each vnode, print a more detailed output. See the **show vnode** command for more information.

callout
A synonym for the **show all callout** command.

ps [**/anow**]
A synonym for **show all procs**.

mac[**hine**] *subcommand* [*args ...*]
Perform a platform-specific command.

The following commands are supported by multiprocessor kernels on these platforms: amd64, i386, macppc, mips64, and sparc64.

cpuinfo
Display the state of each CPU.
ddbcpu *N*
Stop the current CPU and start handling **ddb** on the selected CPU.
startcpu [*N*]
Resume normal processing on the selected CPU, or all CPUs if none is specified.
stopcpu [*N*]
Stop normal processing on the selected CPU, or all CPUs (except the one handling **ddb**) if none is specified.

Other platform-specific commands:

arm:

frame *addr*
Display the trapframe at *addr*.

i386:

sysregs
Display the contents of the privileged registers: *IDTR*, *GDTR*, *LDTR*, *TR*, *CR0*, *CR2*, *CR3*, and *CR4*.

m88k:

ddbcpu *N*
Stop the current CPU and start handling **ddb** on the selected CPU.
frame *addr*
Display the trapframe at *addr*.
regs
Display the registers from when **ddb** was entered.
searchframe [*addr*]
Search for and display stack exception frames, starting from *addr* if given, else the address in register *r31*, and stopping at the next 8k boundary.
where
Display where the current CPU was stopped.

mips64:

tlb [**/p** *asid*] [**/c**] [*tlb*]
trap *??*

sh:

cache [*addr*]
Display the cache, starting from *addr*, defaulting to 0.
frame
Display the switch and trap frames.
tlb Display the TLB.

sparc64:

ctx Display the context addresses for all threads.

VARIABLES

ddb denotes registers and variables by **$***name*. Register names can be found with the **show registers** command.

Some variable names are suffixed with numbers, and some may have a modifier following a colon immediately after the variable name. For example, register variables can have the '`:u`' modifier to indicate a user register (e.g., '`$eax:u`').

Built-in debugger variables currently supported are:

$radix
Input and output radix.
$maxoff
Addresses are printed as *symbol*+*offset* unless *offset* is greater than *$maxoff*.
$maxwidth
The width of the displayed lines.
$lines
The number of lines to page. This is used by the "more" feature.
$tabstops
Tab stop width.
$log
Controls whether the output of **ddb** will also appear in the system message buffer.

These variables can also be controlled outside **ddb** via the 'ddb' sysctl(8) hierarchy.

EXPRESSIONS

Almost all expression operators in C are supported except for '~', '^', and unary '&'. Special rules for expressions in **ddb** are:

identifier
The name of a symbol. It is translated to the address (or value) of the symbol. '`.`' and '`:`' can be used in the identifier. The following can be accepted as an identifier, if supported by an object format dependent routine:
[*filename*`:`]*func*[:*linenumber*]
[*filename*:]*variable*
filename[:*linenumber*]
The symbol may be prefixed with '*symboltablename*::' (e.g., '`emulator::mach_msg_trap`') to specify other than kernel symbols.
number
The radix is determined by the first two letters: '`0x`': hex, '`0o`': octal, '`0t`': decimal, otherwise, the value of *$radix* is used.
`.` *dot*: the current address.
`+` *next*: the next address.
`..` The address of the start of the last line examined. Unlike *dot* or *next*, this is only changed by the **examine** or **write** command.
`'` The last address explicitly specified.
`$`*variable*
The value of a register or variable. The name may be followed by a '`:`' and modifiers as described above with *identifier*.
expr `#` *expr*
A binary operator which rounds up the left hand side to the next multiple of right hand side.
`*`*expr*
Indirection. It may be followed by a ':' and modifiers as described above.

SEE ALSO

ctfstrip(1), gdb(1), nm(1), witness(4), sysctl.conf(5), hangman(6), crash(8), sysctl(8), extent(9), pool(9), uvm_init(9)

HISTORY

This kernel facility first appeared in the MACH 2 operating system developed by CMU. Hangman (which stands for "hangs maniacal analyzer") first appeared in OpenBSD 1.2.