NAME

witness - lock validation facility

SYNOPSIS

options WITNESS
options WITNESS_COUNT
options WITNESS_KDB
options WITNESS_NO_VNODE
options WITNESS_SKIPSPIN

DESCRIPTION

The **witness** module keeps track of the locks acquired and released by each thread. It also keeps track of the order in which locks are acquired with respect to each other. Each time a lock is acquired, **witness** uses these two lists to verify that a lock is not being acquired in the wrong order. If a lock order violation is detected, then a message is output to the kernel console or log detailing the locks involved and the locations in question. Witness can also be configured to drop into the kernel debugger when an order violation occurs.

The **witness** code also checks various other conditions such as verifying that one does not recurse on a non-recursive lock, or attempt an upgrade on a shared lock held by another thread. If any of these checks fail, then the kernel will panic.

The WITNESS_COUNT kernel option controls the maximum number of **witness** entries that are tracked in the kernel. The maximum number of entries can be queried via the *debug.witness.count* sysctl. It can also be set from the loader(8) via the *debug.witness.count* environment variable.

The WITNESS_NO_VNODE kernel option tells **witness** to ignore locking issues between vnode(9) objects.

The flag that controls whether or not the kernel debugger is entered when a lock order violation is detected can be set in a variety of ways. By default, the flag is off, but if the WITNESS_KDB kernel option is specified, then the flag will default to on. It can also be set from the loader(8) via the *debug.witness.kdb* environment variable or after the kernel has booted via the *debug.witness.kdb* sysctl. If the flag is set to zero, then the debugger will not be entered. If the flag is non-zero, then the debugger will be entered.

The **witness** code can also be configured to skip all checks on spin mutexes. By default, this flag defaults to off, but it can be turned on by specifying the WITNESS_SKIPSPIN kernel option. The flag can also be set via the loader(8) environment variable *debug.witness.skipspin*. If the variable is set to a non-zero value, then spin mutexes are skipped. Once the kernel has booted, the status of this flag can be

examined but not set via the read-only sysctl *debug.witness.skipspin*.

The sysctl *debug.witness.watch* specifies the level of witness involvement in the system. A value of 1 specifies that witness is enabled. A value of 0 specifies that witness is disabled, but that can be enabled again. This will maintain a small amount of overhead in the system. A value of -1 specifies that witness is disabled permanently and cannot be enabled again. The sysctl *debug.witness.watch* can be set via loader(8).

The sysctl *debug.witness.output_channel* specifies the output channel used to display warnings emitted by **witness**. The possible values are ‘console’, indicating that warnings are to be printed to the system console, ‘log’, indicating that warnings are to be logged via log(9), and ‘none’. This sysctl can be set via loader(8).

The **witness** code also provides three extra ddb(4) commands if both **witness** and ddb(4) are compiled into the kernel:

show locks [thread]
Outputs the list of locks held by a thread to the kernel console along with the filename and line number at which each lock was last acquired by the thread. The optional *thread* argument may be either a TID, PID, or pointer to a thread structure. If *thread* is not specified, then the locks held by the current thread are displayed.

show all locks
Outputs the list of locks held by all threads in the system to the kernel console.

show witness
Dump the current order list to the kernel console. The code first displays the lock order tree for all of the sleep locks. Then it displays the lock order tree for all of the spin locks. Finally, it displays a list of locks that have not yet been acquired.

SEE ALSO

ddb(4), loader(8), sysctl(8), mutex(9)

HISTORY

The **witness** code first appeared in BSD/OS 5.0 and was imported from there into FreeBSD 5.0.