

FreeBSD Manual Pages

VNODE(9)

FreeBSD Kernel Developer's Manual

VNODE

NAME

vnode -- internal representation of a file or directory

SYNOPSIS

```
#include <sys/param.h>
#include <sys/vnode.h>
```

DESCRIPTION

The vnode is the focus of all file activity in UNIX. A vnode is described by *struct vnode*. There is a unique vnode allocated for each active file, each current directory, each mounted-on file, text file, the root.

Each vnode has three reference counts, *v_usecount*, *v_holdcnt* and *v_writecount*. The first is the number of clients within the kernel who are using this vnode. This count is maintained by [vref\(9\)](#), [vrele\(9\)](#) and [vput\(9\)](#). The second is the number of clients within the kernel who have the recycling of this vnode. This count is maintained by [vhold\(9\)](#) and [vdrop\(9\)](#). When both the *v_usecount* and the *v_holdcnt* of a vnode reach zero then the vnode will be put on the freelist and may be reused for another file, possibly in another file system. The transition from the freelist is handled by [getnewvnode\(9\)](#). The third is a count of the number of clients which are writing into the file. It is maintained by the [open\(2\)](#) and [close\(2\)](#) system calls.

Any call which returns a vnode (e.g., [vget\(9\)](#), [VOP_LOOKUP\(9\)](#), etc.) will increase the *v_usecount* of the vnode by one. When the caller is finished with the vnode, it should release this reference by calling [vrele\(9\)](#) ([vput\(9\)](#) if the vnode is locked).

Other commonly used members of the vnode structure are *v_id* which is used to maintain consistency in the name cache, *v_mount* which points at the file system which owns the vnode, *v_type* which contains the type of object the vnode represents and *v_data* which is used by file systems to store file system specific data with the vnode. The *v_op* field is used by the *VOP_** macros to call functions in the file system which implement the vnode's functionality.

VNODE TYPES

VNON No type.

VREG A regular file; may be with or without VM object backing. If y

want to make sure this get a backing object, call **vnode_create_vobject()**.

VDIR A directory.

VBLK A block device; may be with or without VM object backing. If y want to make sure this get a backing object, call **vnode_create_vobject()**.

VCHR A character device.

VLNK A symbolic link.

VSOCK A socket. Advisory locking will not work on this.

VFIFO A FIFO (named pipe). Advisory locking will not work on this.

VBAD Indicates that the vnode has been reclaimed.

IMPLEMENTATION NOTES

VFIFO uses the "struct fileops" from `/sys/kern/sys_pipe.c`. VSOCK uses the "struct fileops" from `/sys/kern/sys_socket.c`. Everything else use the one from `/sys/kern/vfs_vnops.c`.

The VFIFO/VSOCK code, which is why "struct fileops" is used at all, is artifact of an incomplete integration of the VFS code into the kernel.

Calls to [malloc\(9\)](#) or [free\(9\)](#) when holding a **vnode** interlock, will cau a LOR (Lock Order Reversal) due to the intertwining of VM Objects and Vnodes.

SEE ALSO

[malloc\(9\)](#), [VFS\(9\)](#), [VOP_ACCESS\(9\)](#), [VOP_ACLCHECK\(9\)](#), [VOP_ADVISE\(9\)](#), [VOP_ADVLOCK\(9\)](#), [VOP_ALLOCATE\(9\)](#), [VOP_ATTRIB\(9\)](#), [VOP_BWRITE\(9\)](#), [VOP_CREATE\(9\)](#), [VOP_FSYNC\(9\)](#), [VOP_GETACL\(9\)](#), [VOP_GETEXTATTR\(9\)](#), [VOP_GETPAGES\(9\)](#), [VOP_INACTIVE\(9\)](#), [VOP_IOCTL\(9\)](#), [VOP_LINK\(9\)](#), [VOP_LISTEXTATTR\(9\)](#), [VOP_LOCK\(9\)](#), [VOP_LOOKUP\(9\)](#), [VOP_OPENCLOSE\(9\)](#), [VOP_PATHCONF\(9\)](#), [VOP_PRINT\(9\)](#), [VOP_RDWR\(9\)](#), [VOP_READDIR\(9\)](#), [VOP_READLINK\(9\)](#), [VOP_REALLOCBLKS\(9\)](#), [VOP_REMOVE\(9\)](#), [VOP_RENAME\(9\)](#), [VOP_REVOKE\(9\)](#), [VOP_SETACL\(9\)](#), [VOP_SETEXTATTR\(9\)](#), [VOP_STRATEGY\(9\)](#), [VOP_VPTOCNP\(9\)](#), [VOP_VPTOFH\(9\)](#).

AUTHORS

This manual page was written by Doug Rabson.