

**NAME**

**namei**, **NDINIT**, **NDFREE** – pathname translation and lookup operations

**SYNOPSIS**

```
#include <sys/param.h>
#include <sys/proc.h>
#include <sys/namei.h>

int
namei(struct nameidata *ndp);

void
NDINIT(struct nameidata *ndp, u_long op, u_long flags, enum uio_seg segflg,
        const char *namep, struct thread *td);

void
NDFREE(struct nameidata *ndp, const uint flags);
```

**DESCRIPTION**

The **namei** facility allows the client to perform pathname translation and lookup operations. The **namei** functions will increment the reference count for the vnode in question. The reference count has to be decremented after use of the vnode, by using either **vrelease(9)** or **vput(9)**, depending on whether you specified the **LOCKLEAF** flag or not.

The **NDINIT()** function is used to initialize **namei** components. It takes the following arguments:

*ndp*     The *struct nameidata* to initialize.

*op*     The operation which **namei()** will perform. The following operations are valid: **LOOKUP**, **CREATE**, **DELETE**, and **RENAME**. The latter three are just setup for those effects; just calling **namei()** will not result in **VOP\_RENAME()** being called.

*flags*   Operation flags. Several of these can be effective at the same time.

*segflg*   UIO segment indicator. This indicates if the name of the object is in userspace (**UIO\_USERSPACE**) or in the kernel address space (**UIO\_SYSSPACE**).

*namep*   Pointer to the component's pathname buffer (the file or directory name that will be looked up).

*td*     The thread context to use for **namei** operations and locks.

**NAMEI OPERATION FLAGS**

The **namei()** function takes the following set of “operation flags” that influence its operation:

**LOCKLEAF**   Lock vnode on return. This is a full lock of the vnode; you'll have to use **VOP\_UNLOCK(9)** to release the lock (or use **vput(9)** to release the lock and do a **vrelease(9)**, all in one).

**LOCKPARENT** This flag lets the **namei()** function return the parent (directory) vnode, *ni\_dvp* in locked state, unless it is identical to *ni\_vp*, in which case *ni\_dvp* is not locked per se (but may be locked due to **LOCKLEAF** ). If a lock is enforced, it should be released using **vput(9)** or **VOP\_UNLOCK(9)** and **vrelease(9)**.

**WANTPARENT** This flag allows the **namei()** function to return the parent (directory) vnode in an unlocked state. The parent vnode must be released separately by using **vrelease(9)**.

**NOCACHE**    Avoid **namei()** creating this entry in the namecache if it is not already present. Normally **namei()** will add entries to the name cache if they're not already there.

FOLLOW	With this flag, <b>namei()</b> will follow the symbolic link if the last part of the path supplied is a symbolic link (i.e., it will return a vnode for whatever the link points at, instead for the link itself).
NOOBJ	Do not call <b>vfs_object_create()</b> for the returned vnode, even though it meets required criteria for VM support.
NOFOLLOW	Do not follow symbolic links (pseudo). This flag is not looked for by the actual code, which looks for FOLLOW. NOFOLLOW is used to indicate to the source code reader that symlinks are intentionally not followed.
SAVENAME	Do not free the pathname buffer at the end of the <b>namei()</b> invocation, instead free it later in <b>NDFREE()</b> so that the caller may access the pathname buffer. See below for details.
SAVESTART	Retain an additional reference to the parent directory; do not free the pathname buffer. See below for details.

## ALLOCATED ELEMENTS

The *nameidata* structure is composed of the following fields:

<i>ni_startdir</i>	<p>In the normal case, this is either the current directory or the root. It is the current directory if the name passed in doesn't start with / and we have not gone through any symlinks with an absolute path, and the root otherwise.</p> <p>In this case, it is only used by <b>lookup()</b>, and should not be considered valid after a call to <b>namei()</b>. If SAVESTART is set, this is set to the same as <i>ni_dvp</i>, with an extra <i>vref(9)</i>. To block <b>NDFREE()</b> from releasing <i>ni_startdir</i>, the NDF_NO_STARTDIR_RELE can be set.</p>
<i>ni_dvp</i>	Vnode pointer to directory of the object on which lookup is performed. This is available on successful return if LOCKPARENT or WANTPARENT is set. It is locked if LOCKPARENT is set. Freeing this in NDFREE can be inhibited by NDF_NO_DVP_RELE, NDF_NO_DVP_PUT, or NDF_NO_DVP_UNLOCK (with the obvious effects).
<i>ni_vp</i>	<p>Vnode pointer to the resulting object, NULL otherwise. The <i>v_usecount</i> field of this vnode is incremented. If LOCKLEAF is set, it is also locked.</p> <p>Freeing this in <b>NDFREE()</b> can be inhibited by NDF_NO_VP_RELE, NDF_NO_VP_PUT, or NDF_NO_VP_UNLOCK (with the obvious effects).</p>
<i>ni_cnd.cn_pnbuf</i>	<p>The pathname buffer contains the location of the file or directory that will be used by the <b>namei</b> operations. It is managed by the <i>uma(9)</i> zone allocation interface. If the SAVESTART or SAVENAME flag is set, then the pathname buffer is available after calling the <b>namei</b> function.</p> <p>To only deallocate resources used by the pathname buffer, <i>ni_cnd.cn_pnbuf</i>, then NDF_ONLY_PNBUF flag can be passed to the <b>NDFREE()</b> function. To keep the pathname buffer intact, the ND_NO_FREE_PNBUF flag can be passed to the <b>NDFREE()</b> function.</p>

## FILES

`src/sys/kern/vfs_lookup.c`

**SEE ALSO**

uio(9), uma(9), VFS(9), vnode(9), vput(9), vref(9)

**HISTORY**

This manual page was first written by Eivind Eklund <eivind@FreeBSD.ORG>.

**AUTHORS**

This manual page is rewritten by Hiten M. Pandya <hmp@FreeBSD.ORG>

**BUGS**

The LOCKPARENT flag does not always result in the parent vnode being locked. This results in complications when the LOCKPARENT is used. In order to solve this for the cases where both LOCKPARENT and LOCKLEAF are used, it is necessary to resort to recursive locking.