**Directory Entry Cache (dcache)**

**struct dentry\_operations**

This describes how a filesystem can overload the standard dentry operations. Dentries and the dcache are the domain of the VFS and the individual filesystem implementations. Device drivers have no business here. These methods may be set to NULL, as they are either optional or the VFS uses a default. As of kernel 2.6.22, the following members are defined:

Это описывает, как файловая система может перегрузить стандартные операции dentry. Dentries и dcache являются областью реализации VFS и отдельных файловых систем. Драйверам устройств здесь нечего делать. Эти методы могут иметь значение NULL, поскольку они либо необязательны, либо VFS использует значение по умолчанию. В ядре 2.6.22 определены следующие члены:

**struct** **dentry\_operations** {

int (\*d\_revalidate)(**struct** **dentry** \*, unsigned int);

int (\*d\_weak\_revalidate)(**struct** **dentry** \*, unsigned int);

int (\*d\_hash)(**const** **struct** **dentry** \*, **struct** **qstr** \*);

int (\*d\_compare)(**const** **struct** **dentry** \*,

unsigned int, **const** char \*, **const** **struct** **qstr** \*);

int (\*d\_delete)(**const** **struct** **dentry** \*);

int (\*d\_init)(**struct** **dentry** \*);

void (\*d\_release)(**struct** **dentry** \*);

void (\*d\_iput)(**struct** **dentry** \*, **struct** **inode** \*);

char \*(\*d\_dname)(**struct** **dentry** \*, char \*, int);

**struct** **vfsmount** \*(\*d\_automount)(**struct** **path** \*);

int (\*d\_manage)(**const** **struct** **path** \*, bool);

**struct** **dentry** \*(\*d\_real)(**struct** **dentry** \*, **const** **struct** **inode** \*);

};

**d\_revalidate**

called when the VFS needs to revalidate a dentry. This is called whenever a name look-up finds a dentry in the dcache. Most local filesystems leave this as NULL, because all their dentries in the dcache are valid. Network filesystems are different since things can change on the server without the client necessarily being aware of it.

This function should return a positive value if the dentry is still valid, and zero or a negative error code if it isn’t.

d\_revalidate may be called in rcu-walk mode (flags & LOOKUP\_RCU). If in rcu-walk mode, the filesystem must revalidate the dentry without blocking or storing to the dentry, d\_parent and d\_inode should not be used without care (because they can change and, in d\_inode case, even become NULL under us).

If a situation is encountered that rcu-walk cannot handle, return -ECHILD and it will be called again in ref-walk mode.

вызывается, когда VFS нуждается в повторной проверке dentry. Это вызывается всякий раз, когда поиск имени находит dentry в dcache. Большинство локальных файловых систем оставляют это значение как NULL, потому что все их дентрии в dcache. Сетевые файловые системы отличаются друг от друга, что на сервере все может измениться без того, чтобы обязательно клиент знал об этом.

Эта функция должна возвращать положительное значение, если dentry все еще действителен, и ноль или отрицательный код ошибки, если это не так.

d\_revalidate может быть вызван в режиме rcu-walk (flags & LOOKUP\_RCU). Если в режиме rcu-walk файловая система должна быть повторно проверена, без блокировки или сохранения в dentry, d\_parent и d\_inode не должны измениться без осторожности (потому что они могут измениться, в случае d\_inode, даже стать НУЛЕВЫМИ под нами).

Если возникает ситуация, при которой возникает ситуация, при которой rcu-walk не может справиться, return-ECHILD и он будет вызван снова в режиме ref-walk.

**\_weak\_revalidate**

called when the VFS needs to revalidate a “jumped” dentry. This is called when a path-walk ends at dentry that was not acquired by doing a lookup in the parent directory. This includes “/”, “.” and “..”, as well as procfs-style symlinks and mountpoint traversal.

In this case, we are less concerned with whether the dentry is still fully correct, but rather that the inode is still valid. As with d\_revalidate, most local filesystems will set this to NULL since their dcache entries are always valid.

This function has the same return code semantics as d\_revalidate.

d\_weak\_revalidate is only called after leaving rcu-walk mode.

**d\_hash**

called when the VFS adds a dentry to the hash table. The first dentry passed to d\_hash is the parent directory that the name is to be hashed into.

Same locking and synchronisation rules as d\_compare regarding what is safe to dereference etc.

вызывается, когда VFS добавляет в хеш-таблицу дентри. Первый dentry, переданный в d\_hash, - это родительский каталог, в который должно быть хешировано имя.

Те же правила блокировки и синхронизации, что и d\_compare, в отношении того, что безопасно для разыменования и т. Д.

**d\_compare**

called to compare a dentry name with a given name. The first dentry is the parent of the dentry to be compared, the second is the child dentry. len and name string are properties of the dentry to be compared. qstr is the name to compare it with.

Must be constant and idempotent, and should not take locks if possible, and should not or store into the dentry. Should not dereference pointers outside the dentry without lots of care (eg. d\_parent, d\_inode, d\_name should not be used).

However, our vfsmount is pinned, and RCU held, so the dentries and inodes won’t disappear, neither will our sb or filesystem module. ->d\_sb may be used.

It is a tricky calling convention because it needs to be called under “rcu-walk”, ie. without any locks or references on things.

Вызывается, чтобы сравнить имя Дентри с заданным именем. Первый дентри является родительским дентри для сравнения, второй - дочерним дентри. len и строка имени - это свойства объекта dentry для сравнения. qstr - это имя для сравнения.

Должен быть постоянным и идемпотентным, по возможности не должен образовывать замки и не должен храниться в зубной полости. Не следует разыменовывать указатели вне dentry без особой осторожности (например, d\_parent, d\_inode, d\_name использовать не следует).

Однако наш vfsmount закреплен, а RCU удерживается, так что dentries и inodes не исчезнут, как и наш sb или модуль файловой системы. -> d\_sb можно использовать.

Это сложное соглашение о вызовах, потому что оно должно вызываться как «rcu-walk», т.е. без замков и ссылок на вещи.

**d\_delete**

called when the last reference to a dentry is dropped and the dcache is deciding whether or not to cache it. Return 1 to delete immediately, or 0 to cache the dentry. Default is NULL which means to always cache a reachable dentry. d\_delete must be constant and idempotent.

вызывается, когда последняя ссылка на dentry отбрасывается и dcache решает, кэшировать его или нет. Верните 1 для немедленного удаления или 0 для кэширования данных dentry. По умолчанию - ПУСТО (NULL), что означает всегда кэшировать доступный объект. d\_delete должен быть постоянным и идемпотентным.

**d\_init**

called when a dentry is allocated

**d\_release**

called when a dentry is really deallocated

вызывается, когда дентри действительно освобождается

**d\_iput**

called when a dentry loses its inode (just prior to its being deallocated). The default when this is NULL is that the VFS calls **[iput()](https://www.kernel.org/doc/html/latest/filesystems/api-summary.html" \l "c.iput" \o "iput)**. If you define this method, you must call **[iput()](https://www.kernel.org/doc/html/latest/filesystems/api-summary.html" \l "c.iput" \o "iput)** yourself

вызывается, когда дентри теряет свой индексный дескриптор (непосредственно перед его освобождением). По умолчанию, когда это значение равно NULL, VFS вызывает iput (). Если вы определяете этот метод, вы должны сами вызывать iput ()

**d\_dname**

called when the pathname of a dentry should be generated. Useful for some pseudo filesystems (sockfs, pipefs, …) to delay pathname generation. (Instead of doing it when dentry is created, it’s done only when the path is needed.). Real filesystems probably dont want to use it, because their dentries are present in global dcache hash, so their hash should be an invariant. As no lock is held, d\_dname() should not try to modify the dentry itself, unless appropriate SMP safety is used. CAUTION : **[d\_path()](https://www.kernel.org/doc/html/latest/filesystems/api-summary.html" \l "c.d_path" \o "d_path)** logic is quite tricky. The correct way to return for example “Hello” is to put it at the end of the buffer, and returns a pointer to the first char. dynamic\_dname() helper function is provided to take care of this.

вызывается, когда должен быть сгенерирован путь к dentry. Полезно для некоторых псевдофайловых систем (sockfs, pipefs,…) для задержки генерации имени пути. (Вместо того, чтобы делать это при создании дентри, это делается только тогда, когда необходим путь.) Реальные файловые системы, вероятно, не хотят его использовать, потому что их данные присутствуют в глобальном хэше dcache, поэтому их хеш должен быть инвариантом. Поскольку блокировка не удерживается, d\_dname () не должна пытаться изменить саму dentry, если не используется соответствующая безопасность SMP. ВНИМАНИЕ: логика d\_path () довольно сложна. Правильный способ вернуть, например, «Hello» - поместить его в конец буфера и вернуть указатель на первый символ. Для этого предоставляется вспомогательная функция dynamic\_dname ().

Example :

**static** char \*pipefs\_dname(**struct** **dentry** \*dent, char \*buffer, int buflen)

{

**return** dynamic\_dname(dentry, buffer, buflen, "pipe:[%lu]",

dentry->d\_inode->i\_ino);

}

**d\_automount**

called when an automount dentry is to be traversed (optional). This should create a new VFS mount record and return the record to the caller. The caller is supplied with a path parameter giving the automount directory to describe the automount target and the parent VFS mount record to provide inheritable mount parameters. NULL should be returned if someone else managed to make the automount first. If the vfsmount creation failed, then an error code should be returned. If -EISDIR is returned, then the directory will be treated as an ordinary directory and returned to pathwalk to continue walking.

If a vfsmount is returned, the caller will attempt to mount it on the mountpoint and will remove the vfsmount from its expiration list in the case of failure. The vfsmount should be returned with 2 refs on it to prevent automatic expiration - the caller will clean up the additional ref.

This function is only used if DCACHE\_NEED\_AUTOMOUNT is set on the dentry. This is set by \_\_d\_instantiate() if S\_AUTOMOUNT is set on the inode being added.

вызывается, когда необходимо пройти через автомонтируемый дентри (необязательно). Это должно создать новую запись монтирования VFS и вернуть ее вызывающему. Вызывающей стороне предоставляется параметр пути, указывающий каталог автомонтирования для описания цели автомонтирования и родительскую запись монтирования VFS для предоставления наследуемых параметров монтирования. NULL должен быть возвращен, если кому-то еще удалось сначала выполнить автоматическое монтирование. Если создание vfsmount не удалось, должен быть возвращен код ошибки. Если возвращается -EISDIR, тогда каталог будет рассматриваться как обычный каталог и будет возвращен в режим pathwalk для продолжения просмотра.

Если возвращается vfsmount, вызывающий попытается смонтировать его в точке монтирования и удалит vfsmount из своего списка истечения срока действия в случае неудачи. Vfsmount должен быть возвращен с 2 ссылками на нем, чтобы предотвратить автоматическое истечение срока действия - вызывающий очистит дополнительную ссылку.

Эта функция используется только в том случае, если для dentry задано DCACHE\_NEED\_AUTOMOUNT. Это устанавливается функцией \_\_d\_instantiate (), если для добавляемого inode установлено значение S\_AUTOMOUNT.

**d\_manage**

called to allow the filesystem to manage the transition from a dentry (optional). This allows autofs, for example, to hold up clients waiting to explore behind a ‘mountpoint’ while letting the daemon go past and construct the subtree there. 0 should be returned to let the calling process continue. -EISDIR can be returned to tell pathwalk to use this directory as an ordinary directory and to ignore anything mounted on it and not to check the automount flag. Any other error code will abort pathwalk completely.

If the ‘rcu\_walk’ parameter is true, then the caller is doing a pathwalk in RCU-walk mode. Sleeping is not permitted in this mode, and the caller can be asked to leave it and call again by returning -ECHILD. -EISDIR may also be returned to tell pathwalk to ignore d\_automount or any mounts.

This function is only used if DCACHE\_MANAGE\_TRANSIT is set on the dentry being transited from.

вызывается, чтобы файловая система могла управлять переходом от dentry (необязательно). Это позволяет autofs, например, задерживать клиентов, ожидающих исследования за «точкой монтирования», позволяя демону пройти мимо и построить там поддерево. 0 должен быть возвращен для продолжения вызывающего процесса. -EISDIR может быть возвращен, чтобы указать pathwalk использовать этот каталог как обычный каталог и игнорировать все, что в нем смонтировано, и не проверять флаг автомонтирования. Любой другой код ошибки полностью прервет путь.

Если параметр «rcu\_walk» имеет значение «истина», то вызывающий абонент выполняет обход в режиме обхода RCU. Спящий режим в этом режиме не разрешен, и вызывающего абонента можно попросить выйти из него и позвонить снова, вернув -ECHILD. -EISDIR также может быть возвращен, чтобы указать pathwalk игнорировать d\_automount или любые монтирования.

Эта функция используется только в том случае, если DCACHE\_MANAGE\_TRANSIT установлен для данти, из которой выполняется переход.

**d\_real**

overlay/union type filesystems implement this method to return one of the underlying dentries hidden by the overlay. It is used in two different modes:

Called from file\_dentry() it returns the real dentry matching the inode argument. The real dentry may be from a lower layer already copied up, but still referenced from the file. This mode is selected with a non-NULL inode argument.

With NULL inode the topmost real underlying dentry is returned.

Each dentry has a pointer to its parent dentry, as well as a hash list of child dentries. Child dentries are basically like files in a directory.

Файловые системы оверлейного / объединенного типа реализуют этот метод для возврата одной из нижележащих дентри, скрытых оверлеем. Он используется в двух разных режимах:

Вызывается из file\_dentry (), он возвращает реальный dentry, соответствующий аргументу inode. Настоящий дентри может быть из нижнего слоя, который уже скопирован, но все еще упоминается в файле. Этот режим выбирается с аргументом inode, отличным от NULL.

С NULL inode возвращается самый верхний реальный базовый dentry.

Каждый дентри имеет указатель на его родительский дентри, а также хэш-список дочерних дентри. Дочерние дентри в основном похожи на файлы в каталоге.

**Directory Entry Cache API**

There are a number of functions defined which permit a filesystem to manipulate dentries:

**dget**

open a new handle for an existing dentry (this just increments the usage count)

**dput**

close a handle for a dentry (decrements the usage count). If the usage count drops to 0, and the dentry is still in its parent’s hash, the “d\_delete” method is called to check whether it should be cached. If it should not be cached, or if the dentry is not hashed, it is deleted. Otherwise cached dentries are put into an LRU list to be reclaimed on memory shortage.

**d\_drop**

this unhashes a dentry from its parents hash list. A subsequent call to dput() will deallocate the dentry if its usage count drops to 0

**d\_delete**

delete a dentry. If there are no other open references to the dentry then the dentry is turned into a negative dentry (the d\_iput() method is called). If there are other references, then **[d\_drop()](https://www.kernel.org/doc/html/latest/filesystems/api-summary.html" \l "c.d_drop" \o "d_drop)** is called instead

**d\_add**

add a dentry to its parents hash list and then calls **[d\_instantiate()](https://www.kernel.org/doc/html/latest/filesystems/api-summary.html" \l "c.d_instantiate" \o "d_instantiate)**

**d\_instantiate**

add a dentry to the alias hash list for the inode and updates the “d\_inode” member. The “i\_count” member in the inode structure should be set/incremented. If the inode pointer is NULL, the dentry is called a “negative dentry”. This function is commonly called when an inode is created for an existing negative dentry

**d\_lookup**

look up a dentry given its parent and path name component It looks up the child of that given name from the dcache hash table. If it is found, the reference count is incremented and the dentry is returned. The caller must use dput() to free the dentry when it finishes using it.