Rozdział 1

Algorytm pobrania i-węzła z pamięci iget

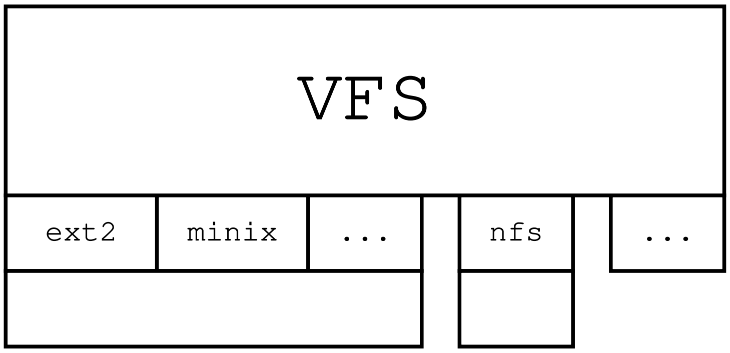
Jacek Fedoryński, Grzegorz W. Fedoryński

Opisano działanie algorytmu pobrania i-węzła z pamięci zawartego w funkcji `iget()` w jądrze Linuksa[1] w wersji 2.2.17.

1.1 Wprowadzenie

1.1.1 VFS

Aby umożliwić obsługę różnych **systemów plików** (ang. *filesystem*), a jednocześnie nie wymagać od modułów z wyższych poziomów wiedzy o ich istnieniu i różnicach między różnymi systemami w jądrze Linuksa wprowadzono warstwę **VFS** (*Virtual File System* lub też *Virtual Filesystem Switch*) zapewniającą jednolity interfejs wyższym warstwom.



Rysunek 1.1: System plików

[1]Linux jest znakiem zastrzeżonym Linusa Torvaldsa

VFS utrzymuje struktury danych oraz udostępnia funkcje, dzięki którym możliwe jest zamontowanie w jednym drzewie różnych systemów plików (nie koniecznie związanych z fizycznymi plikami na dysku, jak np. w przypadku systemów `proc` czy `nfs`) w sposób przezroczysty dla użytkownika.

1.1.2 i-węzeł

W uniksowych systemach plików **i-węzeł** (ang. *inode*) jest strukturą związaną z każdym plikiem, w której przechowywane są informacje takie jak typ pliku, właściciel, prawa dostępu, czasy dostępu oraz ilość i umiejscowienie na dysku bloków zawierających dane pliku.

Różne systemy plików mogą przechowywać inne informacje o plikach w różnych formach. VFS, aby zapewnić wyższym warstwom jednakowy interfejs do systemów plików, jednocześnie zachowując informacje charakterystyczne dla danego systemu plików, operuje strukturą, którą nazywamy „i-węzłem w pamięci" (`struct inode` zdefiniowana w `include/linux/fs.h`).

i-węzeł w pamięci, oprócz informacji o pliku, które przechowywane są na dysku zawiera również pola, które mają sens tylko w pamięci, jak licznik odwołań (`i_count`), pola służące umieszczaniu i-węzła na listach, semafory, stan i-węzła (`i_state`) i inne. Istotnym polem jest wskaźnik `i_op`, pokazujący na strukturę zawierającą adresy funkcji wykonujących operacje na pliku (np. `read`, `write`), charakterystycznych dla danego systemu plików i typu pliku.

Jeśli dany system plików nie przechowuje na dysku żadnych własnych i-węzłów, albo w ogóle nie jest powiązany z żadnymi plikami na dysku, to na potrzeby warstwy VFS w jakiś sposób generuje informacje zawarte w i-węźle w pamięci.

```
struct inode {
        struct list_head        i_hash;
        struct list_head        i_list;
        struct list_head        i_dentry;

        unsigned long           i_ino;
        unsigned int            i_count;
        kdev_t                  i_dev;
        umode_t                 i_mode;
        nlink_t                 i_nlink;
        uid_t                   i_uid;
        gid_t                   i_gid;
        kdev_t                  i_rdev;
        off_t                   i_size;
        time_t                  i_atime;
        time_t                  i_mtime;
        time_t                  i_ctime;
        unsigned long           i_blksize;
        unsigned long           i_blocks;
        unsigned long           i_version;
        unsigned long           i_nrpages;
        struct semaphore        i_sem;
        struct semaphore        i_atomic_write;
        struct inode_operations *i_op;
```

```
        struct super_block      *i_sb;
        struct wait_queue       *i_wait;
        struct file_lock        *i_flock;
        struct vm_area_struct   *i_mmap;
        struct page             *i_pages;
        struct dquot            *i_dquot[MAXQUOTAS];

        unsigned long           i_state;

        unsigned int            i_flags;
        unsigned char           i_pipe;
        unsigned char           i_sock;

        int                     i_writecount;
        unsigned int            i_attr_flags;
        __u32                   i_generation;
        union {
                struct pipe_inode_info          pipe_i;
                struct minix_inode_info         minix_i;
                struct ext2_inode_info          ext2_i;
                ...
                void                            *generic_ip;
        } u;
};
```

1.1.3 Superblok

Podobnie jak i-węzeł jest związany z plikiem, **superblok** (ang. *superblock*) jest strukturą związaną z systemem plików. Analogicznie mówi się o superbloku w pamięci (`struct super_block` w pliku `include/linux/fs.h`) i superbloku na dysku. Podobnie superblok w pamięci oprócz informacji dotyczących konkretnego zamontowanego systemu plików zawiera pola mające sens tylko w pamięci, m.in. wskaźnik do struktury z adresami funkcji wykonujących operacje na tym konkretnym typie systemu plików (`s_op`).

W pamięci znajduje się superblok dla każdego zamontowanego systemu plików (nie dla typu systemu plików), więc i-węzeł identyfikowany jest przez podanie superbloku i numeru i-węzła unikalnego tylko w obrębie jednego systemu plików.

1.2 Listy i-węzłów w pamięci

Warstwa VFS utrzymuje kilka list i-węzłów:

- `inode_in_use` – i-węzły używane
- `inode_unused` – i-węzły nieużywane

- `sb->s_dirty` – i-węzły brudne
- `inode_hashtable` – tablica haszująca

Lista i-węzłów brudnych (czyli takich, których stan w pamięci nie zgadza się ze stanem na dysku) jest utrzymywana osobno dla każdego superbloku (każdego zamontowanego systemu plików), co pozwala na szybkie zapisanie brudnych i-węzłów z jednego systemu plików na dysk.

Każdy „normalny" i-węzeł znajduje się na liście i-węzłów używanych albo brudnych, a oprócz tego w tablicy haszującej, co pozwala na szybki dostęp do niego.

Lista i-węzłów nieużywanych tak naprawdę przechowuje wolne miejsca, na które w razie potrzeby będzie można wczytać nowe i-węzły. i-węzły z tej listy nie znajdują się w tablicy haszującej, gdyż nie przechowują informacji, a gdy potrzebujemy nieużywanego i-węzła bierzemy po prostu pierwszy z listy.

1.3 Algorytm iget

Algorytm pobrania i-węzła z pamięci zawarty jest w funkcji `iget()`, która przyjmuje jako parametry wskaźnik do superbloku systemu plików, z którego pochodzi i-węzeł oraz numer żądanego i-węzła. Funkcja zwraca wskaźnik do i-węzła lub `NULL` w przypadku niepowodzenia. Zdefiniowana jest w pliku `fs/inode.c`.

```
struct inode *iget(struct super_block *sb, unsigned long ino)
```

Funkcja `iget` najpierw sprawdza czy żądany i-węzeł nie został już wcześniej wczytany do pamięci – w takim przypadku znajdowałby się w tablicy haszującej (`inode_hashtable`) i wystarczy go w niej znaleźć. Żeby to sprawdzić wywoływana jest funkcja `find_inode()`, która dostaje jako parametry superblok, numer i-węzła oraz listę z tablicy haszującej, na której znajdowałby się szukany i-węzeł gdyby był w pamięci:

```
static struct inode * find_inode(struct super_block * sb,
  unsigned long ino, struct list_head *head)
```

Funkcja `find_inode()` po prostu przegląda daną listę w poszukiwaniu i-węzła i zwraca go jeśli znajdował się na liście, uprzednio zwiększając licznik odwołań `i_count`. W przeciwnym przypadku funkcja zwraca `NULL`.

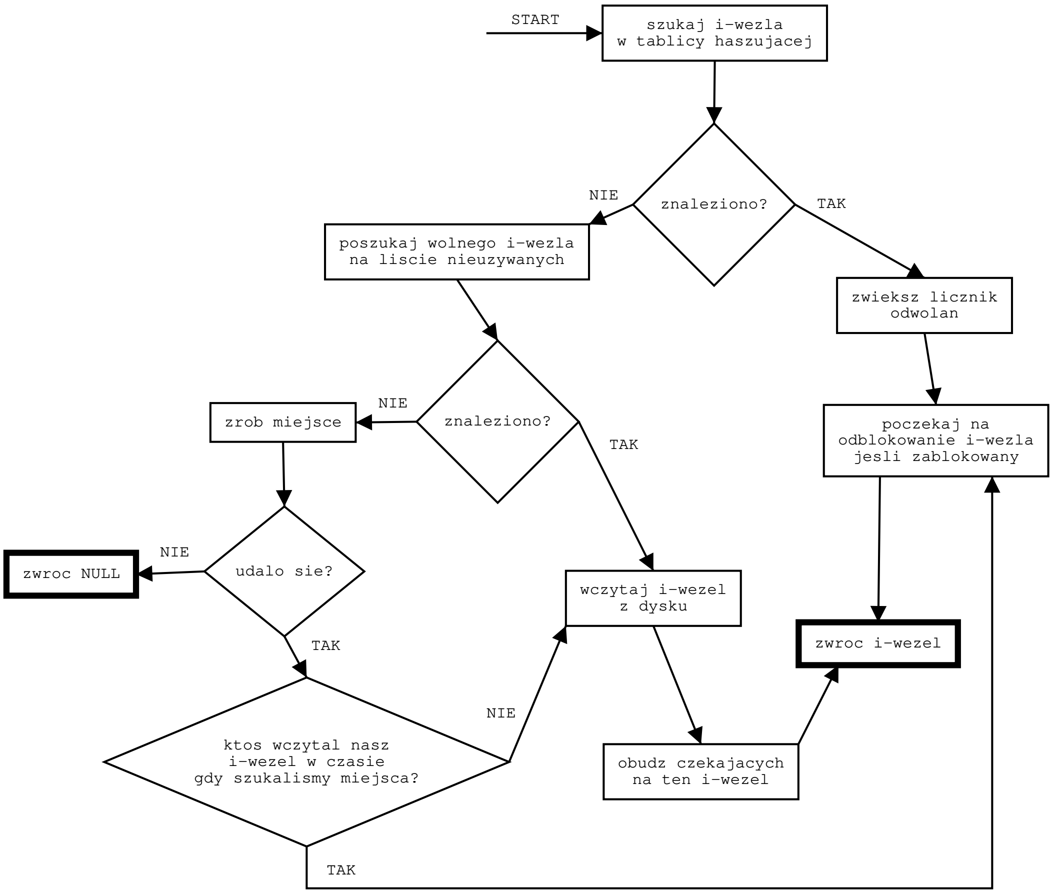
Jeśli i-węzeł był już w pamięci to ewentualnie czekamy aż zostanie odblokowany (funkcja `wait_on_inode()`) i zwracamy go. W przeciwnym przypadku musimy wczytać żądany i-węzeł z dysku. W tym celu wywoływana jest funkcja `get_new_inode()`.

```
static struct inode * get_new_inode(struct super_block *sb,
  unsigned long ino, struct list_head *head)
```

Funkcja ta aby wczytać i-węzeł do pamięci musi znaleźć dla niego miejsce. Sprawdza najpierw czy na liście nieużywanych i-węzłów (`inode_unused`) jest chociaż jeden pusty i-węzeł, w miejsce którego można by wczytać nowy. Jeśli tak to odpowiednim polom i-węzła nadawane są wartości początkowe i wywoływana jest funkcja charakterystyczna dla danego systemu plików fizycznie wczytująca i-węzeł z dysku (`sb->s_op->read_inode()`). Następnie budzone są procesy, które czekają na dany i-węzeł i i-węzeł jest zwracany.

Jeśli lista nieużywanych i-węzłów była pusta to musimy w inny sposób znaleźć miejsce na nowy i-węzeł – wywoływana jest funkcja `grow_inodes()`, która próbuje zaalokować nową pamięć lub wyrzucić z niej mało potrzebne i-węzły. Jeśli to się nie uda, zwracany jest `NULL`, w takim przypadku również `iget()` zwróci `NULL`.

Ponieważ w czasie szukania miejsca wypuszczamy z rąk spinlock, sprawdzamy czy nasz i-węzeł w międzyczasie nie pojawił się w pamięci (`find_inode()`). Jeśli tak to ewentualnie czekamy na jego odblokowanie i zwracamy go. W przeciwnym przypadku wczytujemy fizycznie żądany i-węzeł z dysku tak samo jak poprzednio.



Rysunek 1.2: Algorytm iget

1.4 Skąd wołane jest iget?

Funkcja `iget()` wołana jest z warstwy poniżej VFS, a więc z modułów implementujących konkretne systemy plików, np. z funkcji `ext2_lookup()` z `fs/ext2/namei.c`. Z kolei `iget()` również odwołuje się do tych warstw żeby fizycznie wczytać i-węzeł z dysku, a więc nie jest tak, że warsty niższe nie wiedzą o istnieniu wyższych lub odwrotnie.

1.5 Źródła

- Wszystkie opisywane funkcje zdefiniowane są w pliku `fs/inode.c`.
- Definicje struktur `inode` oraz `super_block` znajdują się w pliku `include/linux/fs.h`.
- Działanie warstwy VFS opisane jest w pliku `Documentation/filesystems/vfs.txt`.
- Często używane w opisywanych funkcjach operacje na listach zdefiniowane są w pliku `include/linux/list.h`.

Nieocenioną pomocą okazała się strona `http://lxr.linux.no/`, na której znajdują się zindeksowane źródła jądra Linuksa.