# Rozdział 1

# Algorytm przekształcania nazwy ścieżkowej w i-węzeł - `namei`

Artur Lewandowski

Rozdział zawiera opis działania algorytmu `namei`, będącego jednym z niskopoziomowych algorytmów jądra, niezbędnym w obsłudze systemu plików. Jest on odpowiedzialny za znajdywanie i-węzła związanego z zadaną nazwą pliku.

## 1.1 Wstęp

W codziennej pracy posługujemy się nazwami plików zrozumiałymi dla człowieka np.

`/usr/local/games/quake2/quake2`

Funkcje systemowe takie jak `open` lub `ln` odwołują się do plików poprzez numer i-węzła. Dlatego potrzebny jest algorytm tłumaczący ścieżkę i nazwę pliku na i-węzeł. Jego ogólne działanie przedstawia **rysunek 1.1**.

Funkcją realizującą to zadanie jest `namei`.

```
struct dentry * __namei(const char *pathname, unsigned int lookup_flags)
```

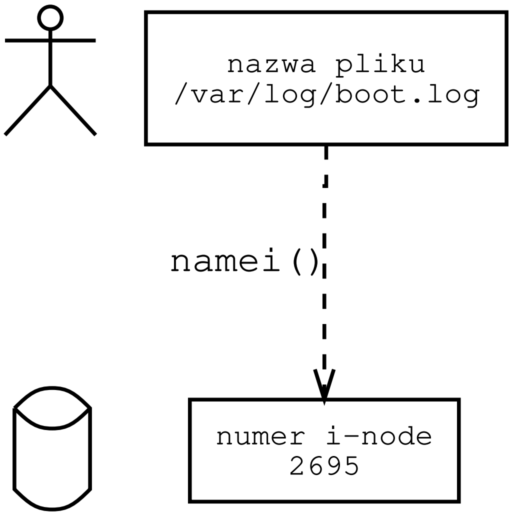
Parametrem funkcji jest ciąg z nazwą ścieżkową, w której kolejne podkatalogi oddzielone są znakiem „/". Wynikiem jest struktura `dentry` wyznaczająca i-węzeł na dysku.

## 1.2 Zasada działania

Ogólny schemat postępowania przedstawia **Algorytm 1**.

## 1.3 Kto woła `namei`?

Proste funkcje – takie jak: ls, chmod – używają podstawowej postaci `namei`. Bardziej skomplikowane funkcje, jak: open, ln – posiadają własne wesje algorytmu – na przykład `open_namei()`




Rysunek 1.1: Ogólne działanie namei

**Algorytm 1** namei()

**Dane wejściowe:** nazwa ścieżkowa pliku

**Wynik:** i-węzeł

```
jeśli nazwa ścieżkowa rozpoczyna się od korzenia „/” to
  i-węzeł roboczy = i-węzeł korzenia; /* algorytm iget */
w przeciwnym przypadku
  i-węzeł roboczy = i-węzeł bieżącego katalogu; /* algorytm iget */
koniec jeśli
dopóki są jeszcze składowe w nazwie ścieżkowej wykonuj
  wczytaj z wejścia kolejną składową nazwy ścieżkowej; /* do znaku „/” */
  sprawdź czy i-węzeł roboczy odpowiada katalogowi i czy prawa dostępu są w porządku;
  jeśli i-węzeł roboczy jest korzeniem i składową jest „..” to
    continue; /* powrót do pętli */
  koniec jeśli
  czytaj katalog (i-węzeł roboczy); /* algorytmy bmap, bread, brelse */
  jeśli składowa pasuje do pozycji w katalogu (i-węzeł roboczy) to
    pobierz numer i-węzła dla pasującej składowej;
    zwolnij i-węzeł roboczy; /* algorytm iput */
    i-węzeł roboczy = i-węzeł pasującej składowej; /* algorytm iget */
  w przeciwnym przypadku
    return(bez i-węzła); /* Składowej nie ma w katalogu */
  koniec jeśli
koniec dopóki
return(i-węzeł roboczy);
```

w pliku `namei.c`. Dodatkowo istnieją dwie wersje: `namei` oraz `lnamei`. Jedyną różnicą między nimi jest to, że `namei` podąża za dowiązaniami a `lnamei` nie. Są one zdefiniowane w pliku `/include/linux/fs.h`.

## 1.4 Gdzie jest `namei`?

Algorytm można znaleźć w źródłach jądra Linux 2.2.17

- `/include/linux/fs.h` definuje makra o nazwach `namei` oraz `lnamei`.

  ```
  #define namei(pathname) __namei(pathname, 1)
  #define lnamei(pathname) __namei(pathname, 0)
  ```

- `/fs/namei.c` zawiera dużo funkcji pomocniczych i sprawdzających oraz realizację algorytmu w funkcjach:
  - `__namei` wykonuje sprawdzenia i wywołuje `lookup_dentry`

    ```
    struct dentry * __namei(const char *pathname, unsigned int lookup_flags)
    ```
  - `lookup_dentry` algorytm `namei` w klasycznej postaci

    ```
    struct dentry * lookup_dentry(const char * name, struct dentry * base,
    unsigned int lookup_flags)
    ```
  - `open_namei` specjalna wersja `namei` dla funkcji `open()`

    ```
    struct dentry * open_namei(const char * pathname, int flag, int mode)
    ```
  - `cached_lookup` zwraca i-węzeł pierwszej składowej podanej ścieżki

    ```
    static struct dentry * cached_lookup(struct dentry * parent,
    struct qstr * name, int flags)
    ```
  - `real_lookup` jak wyżej, ale bez użycia cache'u nazw katalogów

    ```
    static struct dentry * real_lookup(struct dentry * parent,
    struct qstr * name, int flags)
    ```
- `/fs/ext2/namei.c`
- `/fs/vfat/namei.c`
- `/fs/isofs/namei.c` to pliki z dodatkowymi definicjami i funkcjami umożliwiającymi działanie na wszystkich typach podmontowanych systemów plików

## 1.5 Uwagi

1. Algorytm `namei` nie podlegał znaczącym modyfikacjom na przestrzeni lat. Jego idea pozostała nie zmieniona w porównaniu z klasycznym systemem UNIX.
2. Największą zmianą w stosunku do jądra Linux'a 2.0 jest dodanie funkcji `lookup_dentry()` zastępującej funkcje `dir_namei(), _namei(), follow_link()`.

# Literatura

[Bach] Bach M.: *Budowa systemu operacyjnego Unix* Warszawa, WNT 1995.