Rozdział 1

Funkcja iput

Autor: Jerzy Ziemiański.

1.1 Wstęp

Zanim przejdziemy do opisu tytułowej funkcji zwalniającej i-węzeł w pamięci musimy zapoznać się z niezbędnymi pojęciami, takimi jak i-węzeł czy blok specjalny.

Linux obsługuje wiele rodzajów systemów plików, np. ext2, vfat, proc. Możliwe jest to dzięki wprowadzeniu **wirtualnego systemu plików** (*VFS*, czyli *Virtual Filesystem Switch*). Aby jądro potrafiło obsłużyć dany rodzaj systemu plików musi zostać wywołana funkcja `register_filesystem` z parametrem będącym strukturą opisującą jego implementację (`struct file_system_type`). Wśród jej pól jest funkcja `read_super` wywoływana podczas montowania systemu plików tego rodzaju z danego urządzenia. Jej zadaniem jest uzupełnienie (bo część jest już wypełniona przez VFS) **bloku specjalnego** (`struct superblock`) opisującego implementację danego systemu plików. Wynika z tego, że dwa systemy plików tego samego rodzaju mogą być obsługiwane na różne sposoby.

Najbardziej nas interesującą częścią bloku specjalnego jest pole `s_op`, będące wskaźnikiem na strukturę `super_operations`. W skład jej wchodzą wskaźniki do funkcji. Najważniejszymi z nich, z punktu widzenia `iput`, są `put_inode`, `clear_inode` oraz `delete_inode`. Wszystkie one dostają jako parametr wskaźnik do i-węzła (`struct inode`). W VFS-ie wszystkie pliki i katalogi reprezentowane są przez **i-węzły**.

1.2 Struktura i-węzła

I-węzły przechowywane są na urządzeniu zawierającym system plików, np. na dysku lub dyskietce. Poniżej, pisząc o i-węzłach na dysku, mam na myśli i-węzły znajdujące się na dowolnym urządzeniu. Aby jakiś proces mógł z i-węzła korzystać należy go najpierw wczytać do pamięci, oraz, w przypadku modyfikacji, zapisać z powrotem na dysk. I-węzeł w pamięci różni się nieznacznie od tego na dysku, gdyż musi przechowywać dodatkowe informacje, np. ile procesów w danej chwili z niego korzysta.

1.2.1 I-węzeł na dysku

W każdym i-węźle przechowywane są m.in. następujące informacje:

- numer (`i_ino`) - za jego pomocą identyfikowany jest i-węzeł w obrębie danego systemu plików,
- **liczba dowiązań** (`i_nlink`) - do każdego i-węzła może być więcej niż jedno dowiązanie w postaci nazwy. Gdy `i_nlink` osiąga zero - węzeł może zostać usunięty z dysku,

oraz mniej istotne z punktu widzenia `iput` jak rozmiar (`i_size`), właściciel (`i_uid`), grupa (`i_gid`), czasy utworzenia, modyfikacji, dostępu (`i_ctime`, `i_mtime`, `i_atime`).

1.2.2 I-węzeł w pamięci

Poza powyższymi w każdym i-węźle w pamięci przechowywane są m.in.:

- numer logicznego urządzenia systemu plików na którym znajduje się i-węzeł (`i_dev`) - razem z numerem i-węzła (`i_ino`) jednoznacznie identyfikują i-węzeł w całym systemie,
- **licznik otwarć** (`i_count`) - jeśli ma wartość zero to i-węzeł nie jest używany,
- stan (`i_state`) - jest kombinacją bitów:
  - `I_DIRTY` - ustawiony jeśli i-węzeł jest zabrudzony (patrz niżej),
  - `I_LOCK` - wykorzystywany w celu zagwarantowania sobie przez proces wyłącznego dostępu do struktur i-węzła,
  - `I_FREEING` - ustawiony jeśli i-węzeł jest zwalniany,
- dowiązanie do listy na odpowiedniej pozycji w tablicy mieszającej (`i_hash`) - patrz niżej,
- dowiązanie do odpowiedniej listy (`i_list`) - patrz niżej,
- wskaznik (`i_op`) na tablicę wskaźników (`struct inode_operations`) do funkcji implemenetujących podstawowe operacje na i-węźle dla tego systemu plików,
- wskaźnik na blok specjalny (`i_sb`) tego i-węzła.

System przechowuje w pamięci pewną pulę i-węzłów. Każdy z nich może być w jednym z trzech stanów:

- **używany** - i-węzeł jest dokładnym odpowiednikiem i-węzła na dysku,
- **zabrudzony** - i-węzeł jest odpowiednikiem i-węzła na dysku ale został zmodyfikowany (wymaga zapisania na dysk), pole `i_state` ma włączony bit `I_DIRTY`,
- **nieużywany** - i-węzeł w pamięci nie ma odpowiednika na dysku. Równie dobrze mógłby on zostać usunięty z pamięci, jednak nie robi się tego aby uniknąć ciągłego przydzielania i zwalniania pamięci.

Do zarządzania i-węzłami jest wykorzystywana **tablica mieszająca** (`inode_hashtable`), oraz trzy rodzaje **list**:

- **in_use** - zawiera używane i-węzły,
- **dirty** - zawiera zabrudzone i-węzły. Dla każdego zamontowanego systemu pliku istnieje oddzielna lista,
- **unused** - zawiera nieużywane i-węzły.

Do dowiązania i-węzła do jednej z powyższych list służy pole `i_list`. Tablica mieszająca zawiera jedynie i-węzły używane i zabrudzone, do których liczba dowiązań (`i_nlink`) jest większa od zera.

1.3 Algorytm

Funkcja `iput` jest wywoływana przez VFS zawsze wtedy, gdy dany i-węzeł przestaje być procesowi potrzebny, np. gdy zamyka plik.

1.3.1 Zarys

Na początku wywoływana jest funkcja `put_inode` zdefiniowana w bloku specjalnym i-węzła. Następnie zmniejszany jest licznik otwarć i-węzła (`i_count`) i jeśli pozostaje większy od zera lub liczba dowiązań do i-węzła (`i_nlink`) jest większa od zera to `iput` kończy działanie. W przeciwnym razie i-węzeł usuwany jest z dysku przez funkcję `delete_inode` z bloku specjalnego, zwalniana i czyszczona pamięć związaną z zawartością i-węzła, czyszczone jego struktury, uaktualniane przydziały zasobów (quota) oraz stan i-węzła, wywoływana jest funkcja `clear_inode` z bloku specjalnego, a na końcu, i-węzeł przenoszony jest na listę nieużywanych i-węzłów.

1.3.2 Szczegóły

Funkcja `iput` oraz, wykorzystywana przez nią, `clear_inode` zadeklarowane są w pliku `fs/inode.c`.

1.4 Przeszłość i przyszłość

We wcześniejszych jądrach, zgodnie z opisami z poprzednich lat, funkcja `iput` wykonywała więcej zadań. Odpowiedzialna była między innymi za zapisywanie na dysk zabrudzonych i-węzłów, gdy licznik otwarć spadał do zera.

W nowszych jądrach wprowadzono zmiany. Np. w jądrze opatrzonym tym samym numerem ale zaopatrzonym w łatę Mandrake'a dodano obsługę tzw. cieni (ang. shadow). W jądrze 2.4.0 przede wszystkim porządnie skomentowano funkcję (opisywana przez mnie wersja nie posiada żadnych komentarzy). Ponadto dodano wsparcie dla maszyn wieloprocesorowych (SMP) oraz zreorganizowano kod w taki sposób, że jest znacznie czytelniejszy i trochę szybszy (te same czynności nie są już wykonywane kilka razy).

1.5 Zadania

1. Powiedzmy, że lista `unused` zawiera 5 i-węzłów a lista `in_use` 10. Wśród tych dziesięciu jest interesujący nas i-węzeł. Jego pola `i_nlink` oraz `i_count` mają wartości odpowiednio 1 oraz

Algorytm 1 iput

Dane wejściowe: wskaźnik na i-węzeł

Wynik: brak

```
wywołaj funkcję put_inode o ile zdefiniowana dla bloku specjalnego i-węzeła
zablokuj
zmniejsz licznik otwarć (i_count) o jeden
jeśli i_count = 0 to
    jeśli licznik dowiązań (i_nlink) = 0 to
        usuń i-węzeł z tablicy mieszającej
        usuń i-węzeł z listy o ile na jakiejś się znajduje (in_use, dirty lub unused)
        ustaw bit stanu I_FREEING
        wywołaj przy zdjętej blokadzie funkcję delete_inode o ile zdefiniowana dla bloku specjalnego i-węzła
    koniec jeśli
    jeśli i-węzeł nie jest w tablicy mieszającej to
        usuń i-węzeł z listy o ile na jakiejś się znajduje (in_use, dirty lub unused)
        ustaw bit stanu I_FREEING
        wywołaj opisaną poniżej funkcję clear_inode[1]
        dodaj i-węzeł do listy unused oraz zwiększ licznik wolnych i-węzłów
    w przeciwnym przypadku
        jeśli i-węzeł nie jest zabrudzony to
            przenieś go na początek listy in_use
        koniec jeśli
    koniec jeśli
koniec jeśli
jeśli licznik otwarć "przekręcił się" to
    wypisz odpowiedni komunikat
koniec jeśli
odblokuj
```

Algorytm 2 clear_inode

Dane wejściowe: wskaźnik na i-węzeł

Wynik: brak

```
zwolnij strony pamięci związane z zawartością i-węzła
poczekaj na zdjęcie blokady (I_LOCK) z i-węzła (o ile została założona)
jeśli i-węzeł podlega kontroli przydziału zasobów (quota) [2] to
    uaktualnij przydziały zasobów (DQUOT_DROP)
koniec jeśli
wywołaj funkcję clear_inode o ile zdefiniowana dla bloku specjalnego i-węzła
wyczyść bity stanu i-węzła
```

2. Na jakiej liście będzie się znajdował i-węzeł, czy będzie w tablicy mieszającej oraz jakie wartości będą miały pola `i_nlink` oraz `i_count` bezpośrednio po wywołaniu funkcji `iput` ?

2. Rozważmy poprzednie zadanie. Jak będzie się przedstawiać sytuacja gdy funkcja `iput` zostanie wywołana dwa razy pod rząd?

3. Rozwiąż zadania 1. i 2. zakładając, że i-węzeł był na liście `dirty`.

4. Rozwiąż zadanie 1. zakładając, że `i_nlink` miał wartość 0.

5. Rozwiąż zadanie 2. zakładając, że `i_nlink` miał wartość 0.

1.6 Rozwiązania zadań

1. I-węzeł będzie nadal na liście `in_use` i w tablicy mieszającej, `i_nlink` $= 1$, `i_count` $= 1$. Wystarczy zauważyć, że warunek w linii 4 nie będzie spełniony, oraz, że i-węzeł musiał być w tablicy mieszającej (`i_nlink` > 0, używany).

2. Podobnie jak w zadaniu poprzednim, z tym, że `i_count` $= 0$. Tym razem (za drugim przejściem) warunek z linii 4 jest spełniony, ale nadal nie są spełnione warunki z linii 5 i 16. I-węzeł zostanie jedynie przeniesiony na początek listy `in_use`.

3. Rozwiązania podobne do zadań 1. i 2. I-węzeł bedzie na liście `dirty`.

4. I-węzeł będzie nadal na liście `in_use`, `i_nlink` $= 0$ (a co za tym idzie nie będzie w tablicy mieszającej), `i_count` $= 1$. Wystarczy zauważyć, że warunek w linii 4 nie będzie spełniony.

5. I-węzeł będzie na liście `unused`, `i_nlink` $= 0$ (a co za tym idzie nie będzie w tablicy mieszającej), `i_count` $= 0$. Spełnione będą warunki w liniach 4, 5 oraz 11, więc zostaną wykonane linie 7 i 15. Ten i-węzeł nie będzie już nigdy dla nikogo dostępny.