Rozdział 1

Algorytm close

Funkcja close służy do zamykania pliku reprezentowanego przez deskryptor[1]. Poniższy tekst omawia jak działa algorytm w przypadku, gdy plik znajduje się na partycji typu ext2. Rozdział ten został napisany przez Piotrka Kotarbńskiego.

1.1 Struktury danych

W tym podrozdziale omawiam pobieżnie struktury danych, których znajomość jest potrzebna, by zrozumieć jak działa close. Pobieżnie tzn. opisane będą tylko te pola, które są używane przez funkcje. W nawiasach podano pliki gdzie struktury te są zdefiniowane (Są to ścieżki względne w stosunku do katalogu zawierającego źródła jądra)

1.1.1 TASK_STRUCT (include/linux/sched.h)

Struktura ta opisuje proces działający w systemie. Dla nas istotne jest tylko pole files, które jest wskaźnikiem do struktury files_struct zawierającej informacje o plikach otwartych przez ten proces.

1.1.2 FILES_STRUCT (include/linux/sched.h)

Struktura zawiera informacje o plikach otwartych przez pewien proces. Zawiera m. in. następujące pola:

STRUCT FILE **FD (tablica deskryptorów) - jest to tablica wskaźników do struktur file symbolizójących pliki otwarte przez proces.

INT NEXT_FD - najmniejszy wolny deskryptor tego procesu

INT MAX_FDS - największa możliwa wartość deskryptora (wielkość tablicy fd).

Funkcja open zawsze zwraca najmniejszy nie otwarty do tej pory deskryptor. Z tąd to pole, które znacznie usprawnia obsługę deskryptorów.

[1] Jest to liczba naturalna będąca indeksem w tablicy deskryptorów

FD_SET *CLOSE_ON_EXEC - zbiór deskryptorów plików które należy zamknąć w przypadku wywołania funkcji exec przez proces.

FD_SET *OPEN_FDS - zbiór deskryptorów plików otwartych przez proces.

1.1.3 FILE (include/linux/fs.h)

Struktura ta symbolizuje otwarty plik. Interesujące nas pola to:

STRUCT FILE *F_NEXT, **F_PPREV - wskaźniki za pomocą których struktury file połączone są w listę

STRUCT DENTRY *F_DENTRY - wskaźnik na dentry odpowiadające inode'owi(iwęzłowi) odpowiadającemu danemu plikowi

STRUCT FILE_OPERATIONS *F_OP - wskaźnik na strukturę zawierającą wskaźniki operacji plikowych.

MODE_T F_MODE - tryb w jakim otwarty jest plik

UNSIGNED INT F_COUNT - Liczba referencji do tego pliku

1.1.4 FILE_OPERATIONS (include/linux/fs.h)

struktura zawierająca wskaźniki operacji plikowych charakterystycznych dla danego systemu plików. Zawiera m. in. operacje FLUSH i RELEASE. Każda struktura file ma wskaźnik do struktury file_operations odpowiadającej systemowi plików na którym znajduje się dany plik.

1.1.5 DENTRY (include/linux/dcache.h)

w przypadku zwykłego pliku jest to po prostu struktura pośrednia między FILE, a INODE. Posiada ona wskaźnik na DENTRY katalogu macierzystego, liczbę referencji i oczywiście wskaźnik na odpowiedni INODE. Jest tu również wskaźnik do struktury zawierajacej wskaźniki do funkcji charakterystycznych dla danego systemu plików (tak jak w file). W przypadku gdy DENTRY dotyczy katalogu istotne jest jeszcze pole będące wskaźnikiem na listę dentry plików w tym katalogu.

1.1.6 DENTRY_OPERATIONS (include/linux/dcache.h)

operacje na DENTRY charakterystyczne dla systemu plików. Zawiera m. in. metodę D_DELETE.

1.1.7 INODE (include/linux/fs.h)

Struktura opisuje iwęzeł dyskowy wczytany do RAM'u. Zawiera m. in. unię u której polami są opisy iwęzłów charakterystyczne dla danego systemu plików, oraz pole I_WRITECOUNT oznaczające ilu użytkowników pisze do danego pliku.

1.1.8 EXT2_INODE_INFO (include/linux/ext2_fs_i.h)

struktura opisuje inode w systemie plików ext2. Zawiera między innymi informacje o preallokowanych blokach dla danego iwęzła.

1.2 Wyniki działania

Poniższe efekty będą miały miejsce jeśli deskryptor "był otwarty" przez wołający proces.

1. W tablicy deskryptorów procesu w odpowiednim miejscu wstawiany jest nil.
2. Jeżeli dany deskryptor jest mniejszy niż dotychczasowy najmniejszy wolny deskryptor, to jest on aktualizowany.
3. Deskryptor jest usuwany ze zbioru deskryptorów otwartych i tych, które należy zamknąć w przypadku wywołania funkcji exec (o ile tam się znajdował).
4. Wołana jest metoda FLUSH dla danego pliku (w przypadku ext2 jest to operacja pusta).
5. Zdejmowane są ewentualne posix'owe blokady plików, których właścicielem jest wołający użytkownik.
6. Zmniejszany jest licznik referencji do struktury FILE odpowiadającej danemu plikowi.
7. Jeśli licznik ten osiągnie zero to dodatkowo:

- Wywoływana jest metoda RELEASE dla danego pliku (w przypadku ext2 jeśli plik był otwarty do pisania to zwalniane są preallokowane bloki dla inode'a odpowiadającego temu plikowi).
- Jeśli plik był otwarty do pisania to zmniejszane jest pole I_WRITECOUNT inode'a odpowiadającego temu plikowi.
- Zmniejszany jest licznik referencji dla DENTRY odpowiadającego temu plikowi. Jeśli osiągnie on zero to wywoływana jest operacja D_DELETE dla DENTRY (pusta dla ext2). Jeśli dane DENTRY znajduje się na liście dentry'ów katalogu macierzystego to jest ono wstawiane do listy nieużywanych. W przeciwnym razie jest kasowane.
- zdejmowane są ewentualne blokady typu flock z pliku.
- struktura FILE jest usówana z aktualnej listy i wstawiana do listy wolnych.

1.3 Opis sterowania

W tym podrozdziale opiszę dokładnie działanie tzn. jak przebiega sterowanie i co robią poszczególne funkcje. Przy nazwie każdej funkcji umieszczono w nawiasach nazwę pliku, w którym jest ona zdefiniowana.

Wszystko zaczyna się od funkcji SYS_CLOSE (fs/open.c), która jako parametr dostaje deskryptor. Na początek sprawdza czy deskryptor 'jest otwarty' przez proces wołający (wskazuje na niego zmienna globalna TASK_STRUCT CURRENT). W tym celu woła ona funkcję FCHECK (include/linux/file.h), która sprawdza czy deskryptor nie jest większy niż zakres tablicy deskryptorów procesu i zwraca wskaźnik z tej tablicy (który może być nil'em), albo nil jeśli deskryptor przekroczył zakres tablicy deskryptorów. Dalej SYS_CLOSE sprawdza czy wskaźnik ten nie jest nilem. Jeśli tak to wychodzi z błędem i na tym wszystko się kończy.

W przeciwnym wypadku wstawia w tablicy deskryptorów procesu nil w odpowiednim miejscu. Następnie wołana jest funkcja PUT_UNUSED_FD (fs/open.c), która na początku usuwa deskryptor ze zbioru otwartych deskryptorów. Następnie sprawdza czy deskryptor nie jest mniejszy od najmniejszego wolnego deskryptora tego procesu. Jeśli tak to aktualizuje to pole, poczym przekazuje sterowanie z powrotem do SYS_CLOSE.

Tutaj deskryptor jest usuwany ze zbioru tych, które należy zamknąć w przypadku wywołania funkcji exec przez proces. Na koniec wołana jest funkcja FLIP_CLOSE (fs/open.c), która jako parametr dostaje strukturę FILE danego pliku. Wynikiem zwracanym przez SYS_CLOSE jest wynik FLIP_CLOSE.

FLIP_CLOSE po sprawdzeniu pewnych warunków spójności woła metodę FLUSH dla danego pliku. Wynikiem zwracanym przez FLIP_CLOSE (więc także SYS_CLOSE) będzie wynik operacji FLUSH, chyba że jest ona pusta (tzn. odpowiadający jej wskaźnik z FILE_OPERATIONS jest nil'em), wtedy wynikiem jest zero. Następnie zdejmowane są posix'owe blokady z pliku, których właścicielem jest wołający użytkownik poprzez wywołanie funkcji LOCKS_REMOVE_POSIX. Na koniec wołana jest funkcja FPUT (fs/file_table.c), która również dostaje jako parametr strukturę FILE.

FPUT zmniejsza liczbę referncji do tej struktury FILE. Jeśli jest ona nadal większa od zera (ktoś jej jeszcze używa), to na tym kończy się działanie FPUT (a więc zarówno FLIP_CLOSE i SYS_CLOSE).

Natomiast jeśli nikt już nie używa danej struktury FILE, to zdejmowane są z pliku blokady typu flock poprzez wywołanie funkcji LOCKS_REMOVE_FLOCK. Następnie wołana jest funkcja __F_PUT (fs/open.c), której działanie zostanie omówione niżej. Na koniec wołane są funkcje REMOVE_FILP i INSERT_FILE_FREE (obie z fs/file_table.c). Pierwsza z nich usuwa strukturę FILE z aktualnej listy, a druga wstawia ją do listy wolnych (do której mamy dowiązanie poprzez globalną zmienną FILE *FREE_FILPS).

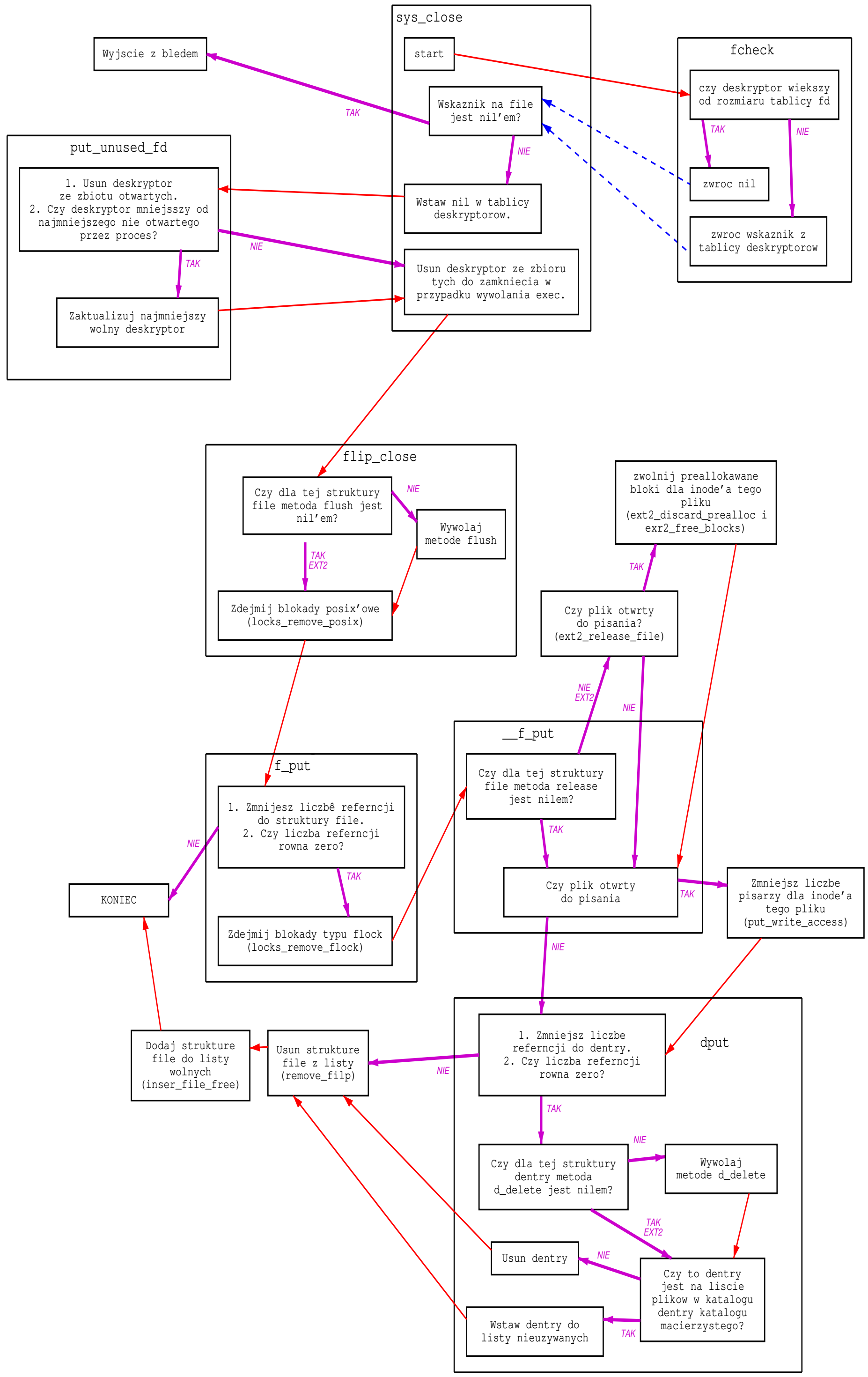
Funkcja __F_PUT woła dla danego pliku metodę RELEASE, która w przypadku ext2 pokazuje na funkcję EXT2_RELEASE_FILE (fs/ext2/file.c). Jeśli plik nie był otwarty do pisania to funkcja ta nic nie robi. W przeciwnym wypadku woła funkcję EXT2_DISCARD_PREALLOC (fs/ext2/inode.c), która zwalnia prealokowane bloki dla inode'a odpowiadającego danemu plikowi (poprzez wywołanie funkcji EXT2_FREE_BLOCKS (fs/ext2/balloc.c)).

Dalej funkcja __F_PUT również sprawdza czy plik był otwarty do pisania. Jeśli tak to wywołuje funkcję PUT_WRITE_ACCESS (fs/namei.c), która zmniejsza pole I_WRITECOUNT inode'a odpowiadającego temu plikowi. Na koniec wołana jest funkcja DPUT (include/linux/dcache.h).

Funkcja ta zmniejsza licznik referencji DENTRY odpowiadającego danemu plikowi. Jeśli spadnie on do zera to wołana jest metoda D_DELETE dla tego DENTRY (pusta w przypadku ext2). Jeśli dane DENTRY znajduje się na liście dentry'ów katalogu macierzystego to jest ono wstawiane do listy nieużywanych. W przeciwnym razie jest kasowane. Na tym kończy się algorytm close.

Diagram przedstawia schemat algorytmu[2]:

[2] bardzo przepraszam za brak polskich czcionek w diagramie, ale zrobiłem go w dia, po czym dowiedziałem się, że program ten ma bug'a nie pozwalającego ich wydrukować.


sys_close
start
Wskaznik na file jest nil'em?
Wstaw nil w tablicy deskryptorow.
Usun deskryptor ze zbioru tych do zamkniecia w przypadku wywolania exec.
Wyjscie z bledem
fcheck
czy deskryptor wiekszy od rozmiaru tablicy fd
zwroc nil
zwroc wskaznik z tablicy deskryptorow
put_unused_fd
1. Usun deskryptor ze zbiotu otwartych. 2. Czy deskryptor mniejsszy od najmniejszego nie otwartego przez proces?
Zaktualizuj najmniejszy wolny deskryptor
flip_close
Czy dla tej struktury file metoda flush jest nil'em?
Wywolaj metode flush
Zdejmij blokady posix'owe (locks_remove_posix)
zwolnij preallokawane bloki dla inode'a tego pliku (ext2_discard_prealloc i exr2_free_blocks)
Czy plik otwrty do pisania? (ext2_release_file)
__f_put
Czy dla tej struktury file metoda release jest nilem?
Czy plik otwrty do pisania
Zmniejsz liczbe pisarzy dla inode'a tego pliku (put_write_access)
f_put
1. Zmnijesz liczbê referncji do struktury file. 2. Czy liczba referncji rowna zero?
Zdejmij blokady typu flock (locks_remove_flock)
KONIEC
dput
1. Zmniejsz liczbe referncji do dentry. 2. Czy liczba referncji rowna zero?
Czy dla tej struktury dentry metoda d_delete jest nilem?
Wywolaj metode d_delete
Usun dentry
Czy to dentry jest na liscie plikow w katalogu dentry katalogu macierzystego?
Wstaw dentry do listy nieuzywanych
Usun strukture file z listy (remove_filp)
Dodaj strukture file do listy wolnych (inser_file_free)
TAK
NIE
TAK EXT2
NIE EXT2