SOI: Laboratorium 6. - System plików
Robert Wojtaś, 283234

Zadanie do zrealizowania

Należy napisać w środowisku systemu Minix program w języku C (oraz skrypt demonstrujący wykorzystanie tego programu) realizujący podstawowe funkcje systemu plików. System plików należy zorganizować w dużym pliku o zadanej wielkości, który będzie "wirtualnym dyskiem". Program powinien tworzyć dysk wirtualny, oraz dokonywać zapisów i odczytów w celu zrealizowania podstawowych operacji na dysku, związanych z zarządzaniem katalogiem, alokacją plików oraz utrzymywaniem unikalności nazw.

Struktura dysku wirtualnego

Dysk wirtualny zgodnie z treścią tworzony był jako plik o dużej, zdefiniowanej przez użytkownika wielkości. Podczas rozwiązywania zadania obmyślono strukturę pliku dysku w następujący sposób:

- **superblok** - zawsze jeden blok, zawierający informacje o liczbie bloków danych w ramach wirtualnego dysku
- **katalog z deskryptorami plików** – rozmiar katalogu zależał od definicji w pliku nagłówkowym VFS.h – MAX_FILECOUNTS. W trakcie demonstracji zadania katalog zajmował 8 bloków I był w stanie przechowywać 1024 deskryptorów plików
- **bitowa mapa zajętości** – tablica bitów przechowująca informacje o zajętości poszczególnych bloków danych, jej rozmiar zależał od ilości bloków danych
- **bloki danych** – podstawowe struktury przechowujące dane, ilość bloków zależała od żądanego rozmiaru dysku

Struktury używane w zadaniu reprezentujące superblok, blok danych, desktyptor plików zostały utworzone w pliku nagłówkowym VFS.h. Rozmiar bloku danych został zdefiniowany jako 4096 bajtów. Bloki danych tworzyły listę, każdy blok otrzymywał czterobajtowy indeks kolejnego bloku, więc użyteczna przestrzeń w danym bloku wynosiła 4092 bajty (FILE_DATA_CHUNK). W pliku tym zostały również utworzone prototypy funkcji wykonujących operacje na pliku dysku.

Zdefiniowane typy danych:

- size_type - 4 bajty, służy do adresowania i wyrażania rozmiaru (w blokach)
- block_index_type – 4 bajty, zamiennik int używany do tworzenia indeksów bloków
- bitmap – typ do tworzenia wskaźników na bitmapy

Struktury VFS (Virtual File System):

- **VFS_superblock** – zajmująca jeden blok informacja o ilości bloków danych w ramach VFS
- **VFS_linked_datablock** – reprezentacja bloku przechowująca tablice bajtów oraz indeks kolejnego bloku
- **VFS_filedescriptor** – struktura deskryptora pliku, rozmiar deskryptora do 32 bajty, w strukturze znajduje się również rozmiar pliku podany w bajtach oraz indeks pierwszego bloku danych dla danego pliku
- **VFS_rootdir** – struktura przechowująca tablicę deskryptorów plików

Operacje na systemie plików:

W ramach zadania ze względu zostały pokryte następujące funkcjonalności systemu plików:

- tworzenie wirtualnego dysku,
- wyświetlanie katalogu dysku wirtualnego,
- usuwanie wirtualnego dysku,
- wyświetlenie zestawienia z aktualną mapą zajętości wirtualnego dysku - czyli listy kolejnych obszarów wirtualnego dysku z opisem: adres, typ obszaru, rozmiar, stan (np. dla bloków danych: wolny/zajęty).

Niestety przez brak czasu nie udało się zaimplementować pozostałych funkcji zdefiniowanych na stronie przedmiotu. Zostały również utworzone polecenia wyższego poziomu, mające za zadanie wykonanie jednej z operacji z powyższej listy. Utworzone programy to:

- **createVFS** – funkcja tworząca plik dysku wirtualnego wykonująca funkcję o tej samej nazwie z biblioteki VFS.h. Jako argumenty należało podać nazwę dysku oraz rozmiar w bajtach.

- **readInfo** – funkcja wypisująca na ekran informacje o dysku, w ramach funkcji bibliotecznej read_VFS_info realizującej wyświetlanie, wykonywane były również funkcje **open_and_read_VFS** oraz **iterate_block_structure.** Informacje wyświetlane w terminalu to:

    - rozmiar bloku danych I użytecznej części bloku danych
    - rozmiar pliku VFS
    - miejsce na dysku wirtualnym
    - liczba bloków danych
    - struktura dysku (informacje o obszarach: superblock, rootdir, mapa bitowa)
    - informacja ile zajmują bloki wolne, a ile zajęte (helpers.h/iterate_block_structure())

- **mls** – funkcja wypisująca listę plików w katalogu, wykonująca funkcję **list_VFS_rootdir,** w ramach której ponownie wykonuje się funkcja open_and_read_VFS()

- **deleteVFS** – funkcja usuwająca plik z dyskiem z użyciem funkcji **unlink()**

Funkcje biblioteczne

Makra I wartości definiowane

W ramach pliku VFS.c zostały utworzone krótkie makra zwracające przykładowo rozmiary elementów struktury dysku, offset w blokach lub bajtach albo obliczające rozmiar pliku z VFS. Są one wykorzystywane w funkcjach np. do obliczenia zajętości dysku. Niektóre przydają się również przy przesuwaniu wskaźnika pliku wirtualnego dysku lub alokowania pamięci (przy odczycie mapy bitowej).

createVFS

Funkcja jako argumenty przyjmuje nazwę pliku dysku oraz rozmiar w bajtach. Na przebieg funkcji składają się operacje otworzenia pliku, wyliczenia ilości bloków potrzebnej do pokrycia zadanej przestrzeni, zapisanie superbloku do pliku oraz ustalenie rozmiaru pliku z VFS. Zwracane 0, gdy utworzenie dysku się powiedzie.

open_and read_VFS

Funkcja otwierajaca plik dysku w celu odczytania informacji o rozmiarze pliku, ilosci superblokow, katalogu glownym VFS. Jako argument przesyłane są: nazwa dysku i wskaźniki na struktury – superblok, katalog dysku, wskaźnik na mapę zajętości, referencje na zmienne przechowujące adres i rozmiar pliku. Funkcja otwiera plik o zadanej nazwie, sprawdza jego rozmiar i dokonuje odczytu poszczególnych obszarów VFS. W razie niepowodzenia zgłaszane są błędy o unikalnych kodach, przetwarzanych na stosowne informacje na wyższym poziomie. Wartości jakie zwraca funkcja to: wskaznik na dysk, wskaznik na superblok, wskaznik na katalog, wskaznik na mape zajetosci, rozmiar dysku.

read_VFS_info

Funkcja wykonująca **open_and_read_VFS,** a następnie odczytująca odpowiednie informacje z otrzymanych struktur. Funkcje oddziałują na siebie poprzez wskaźniki przez co łatwo można dostać się do interesujących nas informacji. Po wykonaniu się funkcji, w przypadku nie wystąpienia błędu na ekran wypisywane są informacje wymienione wyżej w ramach opisu polecenia readInfo. Opis poszczególnych obszarów pamięci zawiera adres (nr bloku, od którego rozpoczyna się dany obszar), rozmiar (w blokach), oraz nazwę danego obszaru np. superblock. W celu wyświetlenia informacji o blokach danych wykonywana jest funkcja **iterate_block_structure** zawarta w pliku **helpers.h.**

- funkcja **iterate_block_structure** zlicza ilość wolnych i zajętych bloków danych. Dokonuje również pogrupowania rodzajów bloków właśnie ze względu na ich zajętość. Dzięki pogrupowaniu I przedstawieniu informacji o ilości bloków zajmowanych przez daną grupę łatwo możemy określić stan zajętości dysku odnosząc się do wcześniej wypisanej liczby wszystkich bloków.

list_VFS_rootdir

Funkcja ta przypomina nieco funkcję opisaną powyżej ze względu na czerpanie informacji o dysku poprzez funkcję open_and_read_VFS. Ponadto, również wypisuje podstawowe informacje o dysku (rozmiar bloku, rozmiar dysku itp.). Po wyświetleniu tych informacji następuje wykonanie pętli przechodzącej przez tablicę deskryptorów plików w strukturze katalogu. W przypadku znalezienia pliku ustawiana jest flaga mówiąca o obecności plików na dysku oraz wypisywane są o pliku: nazwa, rzeczywisty rozmiar, rozmiar na dysku (w kilobajtach). W przypadku gdy tablica jest pusta wyświetlany jest komunikat: “No files.”. Dodatkowym zadaniem funkcji jest zliczanie, w przypadku zapisanych na dysku plików, zajętej pamięci VFS I wyświetlenie informacji o zajętości dysku (wyrażonej w procentach).

Testy

Do wykonania testów należałoby dokończyć zadanie I uzupełnić system plików o pozostałe funkcjonalności. Jednym z testów na pewno byłoby kopiowanie plików z dysku komputera do dysku wirtualnego I na odwrót oraz kontrola poprawnego przesyłu danych np. przez zastosowanie funkcji skrótu przy użyciu polecenia md5sum. Korzystając z funkcji **mls** możliwym byłoby kontrolowanie “dostarczenia” pliku na dysk przez program. Podobnie w przypadku usuwania, kontrola mogłaby przebiegać poprzez obserwowanie katalogu głównego. Należałoby jednak pomyśleć również o teście długotrwałym, w którym przez dłuższy okres wykonywane są operacje kopiowania plików w dwie stronu jak i usuwania plików o różnych rozmiarach.

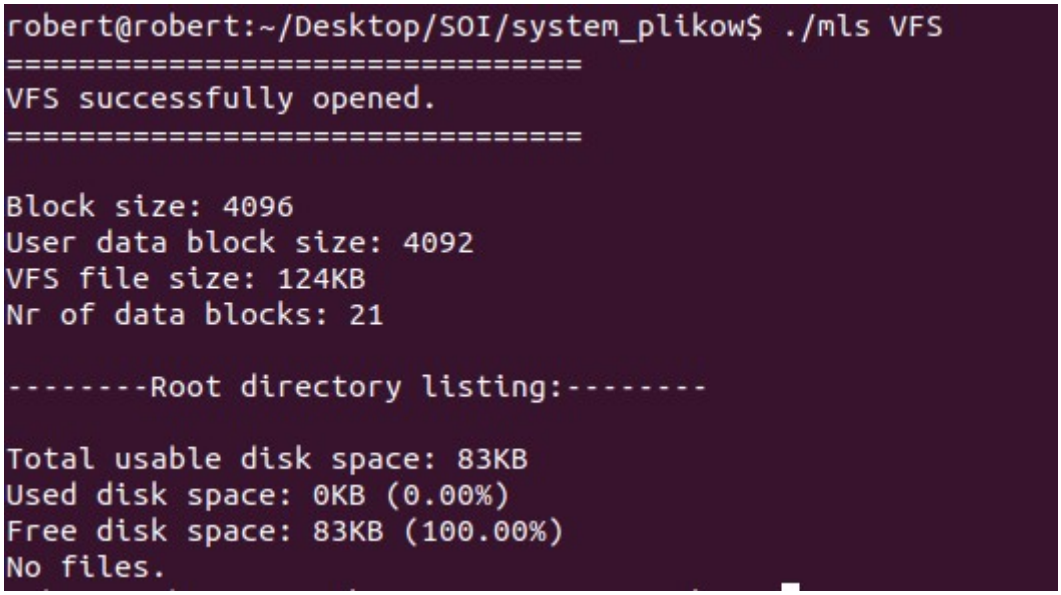
Tworzenie i usuwanie VFS (funkcje ./createVFS oraz ./deleteVFS)

```
robert@robert:~/Desktop/SOI/system_plikow$ ./createVFS VFS 81920
Successfully created VFS in file VFS
robert@robert:~/Desktop/SOI/system_plikow$ ./deleteVFS VFS
Successfully removed file: VFS
robert@robert:~/Desktop/SOI/system_plikow$
```

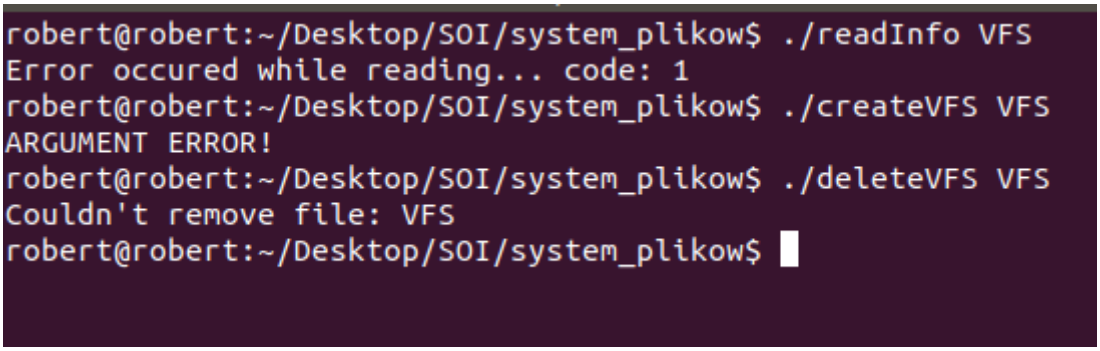
Wyświetlenie informacji o dysku (./readInfo)

```
robert@robert:~/Desktop/SOI/system_plikow$ ./createVFS VFS 81920
Successfully created VFS in file VFS
robert@robert:~/Desktop/SOI/system_plikow$ ./readInfo VFS
================================
VFS successfully opened.
================================

Block size: 4096
User data block size: 4092
VFS file size: 124KB
Usable disk size: 83KB
Nr of data blocks: 21

Disk structure:
-----------------------------------------
addr    size (blocks)           type
-----------------------------------------
0                1    superblock
1                8    root directory
9                1    space usage bitmap
10              21    free data blocks
robert@robert:~/Desktop/SOI/system_plikow$
```

Lista plików katalogu (./mls)



Przykładowe błędy (./createVFS, ./readInfo, ./deleteVFS)



- **readInfo –** kod 1 oznacza, że nie udało się otworzyć pliku o podanej nazwie. Nie było takiego pliku w katalogu.
- **./createVFS –** brak zdefiniowanego rozmiaru pliku
- **./deleteVFS –** nie było możliwe usunięcie pliku nie isteniejącego