Systemy operacyjne – laboratorium

System Plików

Przygotował: Tomasz Bocheński

1. Treść zadania:

Należy napisać w środowisku systemu Minix program w języku C (oraz skrypt demonstrujący wykorzystanie tego programu) realizujący podstawowe funkcje systemu plików.

2. Przyjęte założenia:

System plików zostanie zorganizowany w pliku o zadanej przez użytkownika wielkości, który będzie symulować wirtualny dysk. Pliki na tym dysku znajdować się będą w jednym katalogu (system plików z jednopoziomowym katalogiem). Elementem katalogu będzie opis pliku, który zawierać będzie jego nazwę i wielkość oraz sposób rozmieszczenia pliku na wirtualnym dysku.

3. Przewidywana struktura pliku reprezentującego dysk wirtualny:

Dysk wirtualny podzielony zostanie na kilka części, z których można wyróżnić:

- SuperBlock zawierający takie informacje jak rozmiar dysku, liczba plików, zajęte i wolne miejsce;
- Tablica i-node'ów – każdy węzeł będzie zawierał dane dotyczące jednego pliku jak jego nazwa, rozmiar czy adres początkowego bloku;
- Tablica stanów bloków reprezentująca stany zajęty oraz wolny;
- Dane, podzielone na bloki o określonej długości.

4. Informacje podstawowe o działaniu:

- Zapis sekwencyjny. Próba zapisu będzie podejmowana w pierwszym wolnym fragmencie pamięci. Gdy okaże się on za mały, szukany będzie kolejny wolny fragment.
- Gdy nie zostanie znaleziony odpowiedni fragment pamięci, przeprowadzona zostanie defragmentacja mająca na celu wyeliminowanie „luk" w pamięci.
- Długość bloku nie powinna być zbyt duża, ponieważ będzie to powodowało duże straty w wyniku fragmentacji wewnętrznej.

5. Słabe i mocne strony przyjętego rozwiązania w kontekście zewnętrznej i wewnętrznej fragmentacji:

W kontekście fragmentacji wewnętrznej:
Fragmentacja wewnętrzna będzie zachodzić i jest to nieuniknione, ponieważ dzielimy obszar danych na bloki o określonej długości. Jest to słaba strona tego rozwiązania. Jednak w wyniku zmniejszenia fragmentacji wewnętrznej nie będą stosowane bloki o dużej długości (np. 1024 lub 2048). Jest to mocna strona tego rozwiązania.

W kontekście fragmentacji zewnętrznej:
Fragmentacja zewnętrzna będzie zachodzić i nie da się jej wyeliminować. Jednak przez większość czasu nie będzie ona wpływała na działanie systemu plików. Dopóki jest wystarczająco dużo miejsca dane będą zapisywane na dysku. Dopiero w wyniku zbyt małej ilości miejsca zostanie przeprowadzona defragmentacja. Zatem mocną stroną tego rozwiązania jest fakt, że defragmentację zewnętrzną będzie można eliminować. Jedyny koszt tej eliminacji to wykonanie dodatkowych operacji i to można uznać za słabą stronę tego rozwiązania.

6. Proponowane rozwiązanie:

Zaimplementowane zostaną funkcje:

int createVDisk(int size)

funkcja tworząca wirtualny dysk o podanej jako parameter wielkości, rozmiarze. W wyniku błędu zwracane będzie -1, w wyniku poprawnego stworzenia zwracane będzie 0;

int copyToVDisk(char fileName)*

funkcja kopiująca plik o podanej jako parametr nazwie z dysku systemu Minix na wirtualny dysk. W wyniku błędu zwracane będzie -1, w wyniku poprawnego kopiowania zwracane będzie 0.

int copyFromVDisk(char fileName)*

funkcja kopiująca plik o podanej jako parametr nazwie z dysku wirtualnego na dysk system Minix. W wyniku błędu zwracane będzie -1, w wyniku poprawnego kopiowania zwracane będzie 0.

int showFiles(void)

funkcja wyświetlająca pliki znajdujące się na dysku wirtualnym (wyświetlanie katalogu dysku wirtualnego). Jeśli wyświetlanie zakończy się z błędem zwrócone zostanie -1, jeśli zakończy się sukcesem zwrócone zostanie 0.

int removeFileFromVDisk(char fileName)*

funkcja usuwająca plik o podanej jako parametr nazwie z dysku wirtualnego. W wyniku

sukcesu przy usuwaniu zwracane będzie 0, w wyniku błędu wartością zwracaną będzie -1.

int removeVDisk(void)

funkcja usuwająca wirtualny dysk. Jeśli usuwanie zakończy się powodzeniem, zwracane będzie 0. Jeśli zakończy się błędem to zwrócone zostanie -1.

int showVDiskInfo(void)

wyświetlanie mapy zajętości dysku wirtualnego (czyli listy kolejnych obszarów dysku wirtualnego z opisem: adres, typ obszaru, rozmiar, stan).W wyniku błędu zwracane będzie -1, w wyniku sukcesu zwrócone zostanie 0.

Dodatkowo mogą zostać zaimplementowane jeszcze inne funkcje, zarówno pomocnicze, które pozwolą na implementacje funkcji głównych opisanych powyżej, jak i funkcje pozwalające śledzić działanie dysku i kontrolować poprawność tego działania.

Definicja każdej z funkcji będzie znajdować się w osobnym pliku. Za ich uruchomienie odpowiedzialny będzie kod znajdujący się w pliku main.c.

Deklaracje funkcji, definicje stałych oraz pomocnicze struktury znajdować się będą w pliku nagłówkowym fs.h.

7. Proponowane struktury oraz stałe użyte przy tworzeniu systemu plików:

```
#define BLOCK 1024 (lub 2048)    // rozmiar jednostki bloku
#define MAX_NAME 15              // maksymalna ilość znaków w nazwie pliku



struct vDiskInfo
{
  int size;       // rozmiar dysku wirtualnego
  int filesNmb    // liczba plików na dysku wirtualnym
  int first       // adres pierwszego bloku w katalogu
  int free        // wolne miejsce na dysku wirtualnym
}


struct fileInfo
{
    char name[MAX_NAME];         // nazwa pliku
    int size;                    // rozmiar pliku
    int addr;                    // adres pierwszego bloku pliku
}
```

8. Testowanie

Przygotowany zostanie skrypt , którego zadaniem będzie stworzenie dysku wirtualnego, skopiowanie na niego kilku plików, wypisanie plików znajdujących się na nim, skopiowanie plików z dysku wirtualnego na dysk Minixa, wyświetlenie mapy zajętości dysku, usunięcie kilku plików z dysku wirtualnego i ponowne wyświetlenie mapy zajętości dysku wraz z plikami znajdującymi się na nim. Na końcu dysk zostanie usunięty.

Ponadto oprócz skryptu będzie możliwość własnego testowania / używania przez użytkownika. Będzie on mógł tworzyć dysk wirtualny o zadanym rozmiarze, kopiować pliki między dyskami, usuwać je z dysku wirtualnego a także wypisywać wszystkie pliki lub wyświetlać mapę zajętości dysku.

Pozwoli to na wiarygodne przetestowanie stworzonego systemu plików.