Warszawa, 12.01.2020

student: Patryk Jan Sozański
grupa: 2I5

SYSTEMY OPERACYJNE: LABORATORIUM NR 6
KONCEPCJA

Treść zadania

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.

W pliku na dysku należy zorganizować system plików z jednopoziomowym katalogiem. Elementem katalogu jest opis pliku zawierający co najmniej nazwę, wielkość i sposób rozmieszczenia pliku na wirtualnym dysku.

Należy zaimplementować następujące operacje, dostępne dla użytkownika programu:

1. tworzenie wirtualnego dysku;
2. kopiowanie pliku z dysku systemu Minix na dysk wirtualny;
3. wyświetlanie katalogu dysku wirtualnego;
4. usuwanie pliku z wirtualnego dysku;
5. usuwanie wirtualnego dysku;
6. 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).

Program ma kontrolować wielkość dostępnego miejsca na wirtualnym dysku i pojemność katalogu, reagować na próby przekroczenia tych wielkości. Nie trzeba realizować funkcji otwierania pliku ani czytania/pisania fragmentów pliku. Nie trzeba realizować funkcji związanych z współbieżnym dostępem. Zakłada się dostęp sekwencyjny i wyłączny do wirtualnego dysku.

Należy przygotować demonstrację (zgrupowanie serii poleceń w postaci skryptu interpretera sh) prezentującą słabe i silne strony przyjętego rozwiązania w kontekście ewentualnych zewnętrznej i wewnętrznej fragmentacji.

Sposób rozwiązania problemu

System plików zostanie utworzony zgodnie z założeniami VSFS (Very Simple File System), przedstawionymi w książce *„Operating Systems: Three Easy Pieces”* autorstwa Remzi H. Arpaci-Dusseau i Andrea C. Arpaci-Dusseau.

Wielkość i podział na bloki

Tworzony system plików będzie stworzony na wirtualnym dysku o pojemności 256kB. Zostanie on podzielony na bloki o rozmiarze 4kB, co daje 64 bloki.

Podział funkcjonalny systemu plików User data

Zdecydowaną większość systemu plików będzie stanowiło miejsce na pliki użytkownika. Na tę funkcjonalność zostanie przeznaczonych 56 bloków, z czego pierwszy blok user data będzie miał numer 8.

iNode table

Bloki od 3 do 7 zostaną przeznaczone na tablicę struktur iNode (po polsku i-węzły). Każdy iNode przechowuje informację o pliku takie jak prawa dostępu, data ostatniego dostępu, data ostatniej modyfikacji i przede wszystkim wskaźnik/wskaźniki na blok w regionie user data, w którym znajduje się zawartość pliku.

Sposób implementacji wskaźników może być różny. Najprostszą metodą jest przechowywanie jednego wskaźnika w strukturze iNode na początek pliku i jeśli plik jest większy niż jeden blok, to wskaźnik na następny blok umieszczać na końcu tego bloku. Wielkość iNodów zostanie dobranych tak, aby mogło ich być więcej niż bloków data.

Mapy zajętości bloków

Bloki 1 i 2 przeznaczone zostaną na mapy zajętości tablicy iNodów oraz obszaru user data. Zostaną zrealizowane za pomocą prostych map bitowych, w których bit ustawiony będzie oznaczał zajętość danego bloku, a nieustawiony blok wolny.

Superblock

Na pozycji 0 znajdzie się struktura super bloku. Będzie ona opisem funkcjonalnym całego systemu plików. Przechowywać będzie informację o rozmiarze bloków, liczbie struktur iNode, liczbie bloków danych. Dodatkowo, w strukturze znajdzie się wskaźnik na początek tablicy iNodów oraz początek sekcji danych, a także o rozmiarach poszczególnych struktur.

Program

Tworzenie nowego systemu plików oraz dostęp do niego będzie realizował program fs. Będzie on realizował wszystkie wymienione funkcjonalności opisane w treści polecenia. Aby ułatwić pisanie skryptów testowych, z narzędzia będzie korzystało się podobnie jak z programów typu git, tj. na podstawie argumentu będzie wykonywane polecenie na systemie plików, a następnie zakończy pracę.

Program zostanie w całości napisany w języku C.

Narzędzie będzie realizowało następujące funkcjonalności:

- tworzenie i inicjalizacja wirtualnego dysku w aktualnej lokalizacji,
- kopiowanie podanego pliku z dysku systemu Minix na dysk wirtualny z uwzględnieniem dostępnej pojemności,
- wyświetlanie zawartości katalogu dysku wirtualnego,
- wyświetlanie zawartości podanego pliku,
- edycja wybranego pliku,
- usuwanie wskazanego pliku z wirtualnego dysku,
- usuwanie całego 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).

Testowanie

Testowanie rozwiązania zostanie przeprowadzone w sposób półautomatyczny za pomocą specjalnie przygotowanych skryptów sh.

Tworzenie i usuwanie dysku wirtualnego

Test powinien sprawdzać, czy narzędzie poprawnie tworzy wirtualny dysk o zadanym wcześniej rozmiarze. Po stwierdzeniu poprawności kreacji dysku i jego inicjalizacji, dysk zostanie usunięty i zostanie sprawdzona poprawność usuwania wirtualnego dysku.

Tworzenie i usuwanie plików

Testy powinny również mieć możliwość sprawdzenia poprawności tworzenia plików i ich usuwania. W teście zostanie utworzonych kilkadziesiąt plików w celu sprawdzenia zachowania systemu przy przekroczeniu maksymalnej liczby plików w systemie. Test pozwoli również na sprawdzenie, jak zajmowane i zwalniane są bloki pamięci dyskowej poprzez wyświetlanie zestawienia z aktualną mapą zajętości wirtualnego dysku.

Pisanie, czytanie i kopiowanie

Ważnym testem będzie sprawdzanie poprawności pisania do plików poprzez narzędzie lub kopiowanie plików z systemu MINIX do wirtualnego dysku. Poprawność testu zostanie stwierdzona poprzez odczytanie i porównanie zapisanej zawartości z oryginalną. Możliwe są różne scenariusze tejże operacji m.in. w przypadku, gdy na wirtualnym dysku jest lub nie ma wystarczająco miejsca, by zapisać konkretny plik albo nie będzie dostępny wystarczająco pojemy jeden ciągły fragment pamięci. Ważnym aspektem do przetestowania będzie zachowanie systemu w przypadku zwiększenia rozmiaru pliku już utworzonego w taki sposób, że przestanie on się mieścić w oryginalnej ilości bloków, którą na początku zaalokował (tak zwane pisanie z alokacją).

Parametry last-accessed, last-modified

Każda operacja powinna skutecznie modyfikować parametry pliku. Test powinien za pomocą narzędzia sprawdzać daty ostatniego dostępu i ostatniej modyfikacji pliku za pomocą prostych operacji porównania.

Prezentacja zjawiska fragmentacji

Test powinien alokować miejsce na kilka dużych (zajmujących kilka bloków) i kilka małych plików, następnie usunąć cześć z nich i znowu stworzyć kilka dużych plików tak, aby zaprezentować, jak dany system plików radzi sobie ze zjawiskiem fragmentacji. Podczas tego testu zostanie wykorzystana funkcjonalność wyświetlenia zestawienia z aktualną mapą zajętości wirtualnego dysku.