SPRAWOZDANIE

SOI - Laboratorium

System plików

Dominik Szaciłowski

1.Treść zadania

Należy napisać w środowisku systemu Minix program w języku C (oraz skrypt demonstrujący wykorzystanie tego programu) realizujący podstawowe funkcje systemu plików. Należy zaimplementować następujące operacje, dostępne dla użytkownika programu:

- tworzenie wirtualnego dysku
- kopiowanie pliku z dysku systemu Minix na dysk wirtualny,

- kopiowanie pliku z dysku wirtualnego na dysk systemu Minix,

- wyświetlanie katalogu dysku wirtualnego,

- usuwanie pliku z wirtualnego dysku,

- 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).

2. Głowne założenia:

System pliów został został stworzony na wzór systemu plików FAT. Strefa data została podzielona na 64 bloki po 64B każdy. System plików składa się kolejno z: Superblock, DTF(Dirctory Table Format), FAT(File Allocation Table) oraz data blocks.

```
typedef struct SuperBlock {
        short firstFATBlock;
        short firstDataBlock;
        short sizeOfDisc;               //rozmiar dysku wraz z Superblokiem,DTF oraz FAT
        short numberOfFreeBlocks;
        short firstDTFBlock;
        short a;                        //wolne miejsa dla wyrownania oraz innych ewentualnych
        short b;                          potrzebnych danych
        short c;
}SuperBlock;

typedef struct File {                   //DTF jest to tablica struktury File
        char fileName[MAX_FILENAME_SIZE];
        short startingBlock;
        short fileSize;         /* -1 -> spot is free */
}File;

typedef struct FatRow {                 //FAT jest to tablica struktur FatRow
        short state;
        short next;             /*-1 -> end of file */
}FatRow;
```

3. Utworzone funkcje: (implementacja znajduje sie w pliku FAT.h)

```
void initValues(void);          //służy do inicjowania wartości początkowy w superbloku, DTF oraz
                                  FAT
```

int createDisc(void); //tworzy nowy, pusty dysk. Zwraca ERROR_FILE_CREATING gdy tworzenie sie nie powiedzie
void loadSuperBlock(FILE* disc); //funkcje służące do odczytu z pliku binarnego do tablic
void loadDTF(FILE* disc); w celu dalszych modfikacji lub odczytow
void loadFAT(FILE* disc);

int doesFileExist(char* const name); //funkcja sprawdza czy w DTF juz istnieje plik o podanej nazwie

void actualizeSuperBlock(FILE* disc); //funkcje służące do zapisu tablic do pliku
void actualizeDTF(FILE* disc); binarnego (dysku)
void aztualizeFAT(FILE* disc);

int copyOnDisc(char* const name, char* const name_after); //funkcja kopiująca plik o nazwie name do dysku pod nazwa name_after, zwraca odpowienie błędy w przypadku niepowodzenia. Aktualizuje SuperBlock, DTF oraz FAT. Jesli plik nie miesci sie w jednym bloku, dzieli go i umieszcza kolejno w wolnych na dysku blokach

Int copyFromDisc(char* const name, char* const name_after); //funkcja kopiujaca plik z dysku o nazwie name do minixa pod nazwa name_after, zwraca odpowienie błędy w przypadku niepowodzenia. Aktualizuje SuperBlock, DTF oraz FAT.

Void resetConnected(int DTFPosition, FILE* disc); //funkcja resetuje wszystkie dane zwiazane z plikiem na pozycji DTFPosition w DTF. Uzyta w deleteFile().
Int deleteFile(char* const name); //funkcja usuwa plik o nazwie name z dysku. Czyści również ślad po pliku w data block (błednie z powodu wydłużenia czasu działania funkcji; poczatkowa idea jednak to bylo zapewnienie bezpieczenstwa; nie pozostawienie sladu na dysku)

int showFolder(void); //funkcja pokazuje jakie pliki znajduja sie obecnie w dysku
int resetDisc(void); //format dysku
int deleteDisc(void); //usuniecie dysku
int showInsides(void); // w czytelny sposob wyswietla aktualny stan superblock, DTF oraz FAT w dysku

4. Testy
Dysk można testowac dowolnie z wykorzystaniem pliku Source.c. Po skompilowaniu go poprzez cc Source.c -o Disc mozemy sterowac dyskiem w najstepujacy sposob:
./Disc 0 //utworzenie pustego dysku
./Disc 1 name1 name2 //kopiowanie pliku name1 z minixa do dysku pod nazwa name2
./Disc 2 name1 name2 //kopiowanie pliku name1 z dysku do minixa pod nazwa name2
./Disc 3 //pokazuje obecny stan superbloku, DTF oraz FAT
./Disc 4 //pokazuje pliki znajdujace sie obecnie w dysku
./Disc 5 //resetuje dysk
./Disc 6 //usuwa dysk
./Disc 7 name //usuwa plik name z dysku

Test 1: Przykładowe działanie dysku. Uruchamiamy skrypt c_example.sh. Otrzymujemy na konsoli:

Nastapi skopiowanie 4 razy pliku ex1.txt do dysku pod roznymi nazwami

Nastepnie usuniemy 1. oraz 3. plik
Gdy przez usuniecie powstana dziury,
dodamy nastepny plik tym razem wiele wiekszy
Na samym koncu nastapi skopiowanie najwiekszego pliku
z powrotem na minixa pod inna nazwa
Rozpoczecie
Skopiowano pliki do dysku. Stan mozna zobaczyc w pliku result_2_1.txt
Nastapi zademonstrowanie funkcji
showFolder wyswietlajacej obecne pliki w dysku
ex1.txt ex2.txt ex3.txt ex4.txt
Usuwanie pliku ex1.txt oraz ex3.txt
Zakonczono stan zapisano w pliku result_2_2.txt
showFolder:
ex2.txt ex4.txt
Nastapi skopiowanie pliku long.txt
Zakonczono, Stan dysku mozna zobaczyc w pliku result_2_3.txt
showFolder:
long.txt ex2.txt ex4.txt
Kopiowanie pliku long.txt z dysku na minixa pod nazwa copy_long.txt

Poszczególne stany dysku w tescie:
1. Początkowy:

Rozmiar Dysku: 5904
Ilosc wolnych blockow: 64
Byte, na ktorym zaczyna sie DTF: 16
Byte, na ktorym zaczyna sie FAT: 1552
Byte, na ktorym zaczyna sie DATA: 1808
Directory Table Format //wypelniony poczatkowymi wartosciami
Nazwa: Rozmiar: Pierwszy Blok:
-1 0
-1 0
-1 0
-1 0
.
.
.
-1 0
File Allocation Table
Numer: Status: Nastepny: //Status: 1-Free, 0-FULL
0 1 -1
1 1 -1
2 1 -1
3 1 -1
.
.
.
63 1 -1

2. Po dodaniu czterokrotnie pliku ex1.txt do dysku pod roznymi nazwami

Rozmiar Dysku: 5904

Ilosc wolnych blockow: 52
Byte, na ktorym zaczyna sie DTF: 16
Byte, na ktorym zaczyna sie FAT: 1552
Byte, na ktorym zaczyna sie DATA: 1808
Directory Table Format
Nazwa: Rozmiar: Pierwszy Blok:
ex1.txt 160 0 //plik ex1.txt o rozmiarze 160B oraz rozpoczyna sie on na
ex2.txt 160 3 bloku
ex3.txt 160 6
ex4.txt 160 9
-1 0
.
.
.
-1 0
File Allocation Table
Numer: Status: Nastepny:
0 0 1 //1. blok pliku ex1, nastepny: 1
1 0 2 //2. blok pliku ex1, nastepny: 2
2 0 -1 //3. blok pliku ex1, nastepny:-1 → koniec
3 0 4 //dla plikow ex2,ex3,ex4 analogincznie
4 0 5
5 0 -1
6 0 7
7 0 8
8 0 -1
9 0 10
10 0 11
11 0 -1
12 1 -1
.
.
.
63 1 -1

3. Po usunięciu pliku ex1 oraz ex2:

Rozmiar Dysku: 5904
Ilosc wolnych blockow: 58
Byte, na ktorym zaczyna sie DTF: 16
Byte, na ktorym zaczyna sie FAT: 1552
Byte, na ktorym zaczyna sie DATA: 1808
Directory Table Format
Nazwa: Rozmiar: Pierwszy Blok:
-1 0 //puste miejsca po pliku ex1
ex2.txt 160 3
-1 0 //ex3
ex4.txt 160 9
-1 0
.
.
.

-1 0
File Allocation Table
Numer: Status: Nastepny:
0 1 -1 //zwolnione bloki ex1
1 1 -1
2 1 -1
3 0 4
4 0 5
5 0 -1
6 1 -1 //zwolnione bloki ex3
7 1 -1
8 1 -1
9 0 10
10 0 11
11 0 -1
12 1 -1
.
.
.
63 1 -1

4. Po skopiowaniu pliku long.txt na dysk:

Rozmiar Dysku: 5904
Ilosc wolnych blockow: 51
Byte, na ktorym zaczyna sie DTF: 16
Byte, na ktorym zaczyna sie FAT: 1552
Byte, na ktorym zaczyna sie DATA: 1808
Directory Table Format
Nazwa: Rozmiar: Pierwszy Blok:
long.txt 400 0 //plik long.txt, wiekszy niz pozostale; potrzebne 7 blokow
ex2.txt 160 3
-1 0
ex4.txt 160 9
-1 0
.
.
.
-1 0
File Allocation Table
Numer: Status: Nastepny:
0 0 1 //bloki 0-2 naleza do long, przechodzimy do 6
1 0 2
2 0 6
3 0 4
4 0 5
5 0 -1
6 0 7 //bloki 6-8 naleza do long, przechodzimy do 12
7 0 8
8 0 12
9 0 10
10 0 11

11 0 -1
12 0 -1   //12 blok nalezy do long, koniec
13 1 -1
  .
  .
  .
63 1 -1

Następnie plik zostal skopiowany z dysku do minixa I zostal zapisany do pliku copy_long.txt
Jak mozemy zauwazyc, long.txt(z minixa) oraz copy_long.txt(z dysku) sa takie same.
Zostaly rowniez sprawdzone funkcja cmp.

Test 2:
do pustego dysku probujemy dodac dwa pliki ale chcemy nazwac je tak samo:
./Disc 0
./Disc 1 ex1.txt ex1.txt   //za pierwszym razem sie udaje
./Disc 1 ex1.txt ex1.txt   //taki plik juz istnieje w dysku, wynik:

```
# ./Disc 1 ex1.txt ex1.txt
# ./Disc 1 ex1.txt ex1.txt
File with that name already exist in disc.
*
```

Test 3:
do pustego dysku probujemy dodac plik ktorego rozmiar jest wiekszy od dostepnej pamieci dla przypomnienia dostepna pamiec to: 64*64 = 4096, rozmiar pliku tobig.txt: 4481

```
# ./Disc 0
# ls -l tobig.txt
-rw-r--r-- 1 root  operator  4481 Jan 19 19:18 tobig.txt
# ./Disc 1 tobig.txt tobig.txt
File is larger than space in disc. Couldnt copy a file.
*
```

Test 4:
probujemy skopiowac plik z dysku ktory nie istnieje

```
File is larger than space in disc. Couldnt copy a file.
* ./Disc 0
# ./Disc 2 niematakiego.txt niematakiego.txt
File with such name doesnt exist inside disc
*
```