ADAM STECIUK – SOI – lab6 – RAPORT

Na poprzednich zajęciach prezentowałem już gotowe rozwiązanie problemu, więc koncepcja była finalnym pomysłem. Na czerwono dopisuję wprowadzone poprawki związane z implementacją usuwania plików i testami programu.

Treść zadania

http://www.ia.pw.edu.pl/~tkruk/edu/soib/lab/t6.txt

Propozycja rozwiązania

Przy uruchamianiu programu - użytkownik będzie mógł zdecydować czy chce utworzyć nowy dysk wirtualny czy otworzyć już istniejący. W przypadku pierwszym musi zadeklarować jego wielkość przy uruchamianiu programu. Wielkość musi być większa niż pamięć zajmowana przez inode'y, których ilość (a także maksymalna ilość plików) obsługiwanych przez system zdefiniowana jest jako FILES_NUM. Ze względu na uproszczony charakter tworzonego systemu plików zadanego w zadaniu unikam potrzeby definiowania superblock'u zawierającego informację o adresach pamięci zarezerwowanych na inode'y oraz na pamięci dla plików po przez zapisywanie stałej liczby inode'ów zawsze na początku pamięci wirtualnego dysku. Każdy inode jest struckturą z polami:

- char fileName[MAX_FILE_NAME];
- unsigned short fileSize;
- unsigned short fileBegin;
- short exist;

Tworzona jest również lista inode'ów

- inode data;
- struct node* next;

Struktura pamięci:

Pliki kopiowane na dysk wirtualny są zapisywane w części plikowej w sposób ciągły. Z tego powodu do pierwszego usunięcia pliku z dysku – nie występują dziury w pamięci. Po pojawieniu się dziur, przy kopiowaniu na dysk wirtualny, plik jest zapisywany w pierwszej (zgodnie z kolejnością adresów) znalezionej dziurze o rozmiarze większym, bądź równym wielkości rzeczonego pliku. Jeżeli taka dziura nie istnieje – plik dopisywany jest na samym końcu, tak jak to miało miejsce na początku.

Informacje o każdym następnym pliku są zapisywane do kolejnych inode'ów w liście a przy wychodzeniu z programu i zamykaniu dostępu do wirtualnego dysku zapisywane w pamięci. Rozwiązanie to pozwala na dużo szybsze kopiowanie plików z i na dysk wirtualny, jednakże nie jest ono odporne na awarie (np. nieoczekiwane przerwanie działania programu). Podczas kopiowania pliku z wirtualnego dysku program przeszukuję listę inode'ów w celu odnalezienia tego odpowiadającego podanej przez użytkownika nazwie, odczytuje z niego wielkość i adres początku pliku a następnie przenosi dane do docelowego pliku.

W programie zaimplementowałem następujące funkcje:

- int getLine (char *linia, int max_dl);
- long isEnoughSpace(long size);
- void diskStat(void);
- listNode* prevInode(long end);
- int writeToDisk(int src, char*dest, long size);
- void readFromDisk(void);
- listNode* getListNode(char* name);
- void clearDisk(void);
- int closeDisk(void);
- int openDisk(void);
- void deleteInode(char* del);
- void addInode(listNode* new);
- int clearInodeList(void);
- void deleteInodeList(void);
- int setInodeList(void);
- long usedDisk(void);
- void readInodeList(void);
- int openVFS(void);
- int isUnique(char* name);
- void copyToDisk(void);
- void help(void);
- int createDisk(void);
- int userMenu(void);
- int test01(void);
- int test02(void);
- int copyToDiskTest(int i, int testNum);
- char* concat(const char *s1, const char *s2);

FILES_NUM domyślnie ustawiona wartością 10.

Program został zabezpieczony przed:

- Podaniem nieprawidłowych argumentów startowych
- Przepełnieniem listy inode'ów
- Stworzeniem dwóch plików o takich samych nazwach
- Kopiowaniem z nieistniejących plików
- Przekroczeniem dostępnej pamięci dyskowej
- Używaniem niepoprawnie utworzonego pliku
- Stworzeniem dysku niemieszczącego zadanej liczby inode'ów
- Pracą na niepoprawnie utworzonym dysku
- Pracą na uszkodzonej liście inode'ów

Nowe Testowanie

Dodatkowy test będzie sprawdzał funkcjonalność programu związaną z rozwiązywaniem problemu fragmentacji i zapełnianiem dziur w pamięci. Test będzie się składał z:

- Wkopiowania maksymalnej możliwej ilości plików 1,5kB
- Usunięciu co drugiego pliku
- Wkopiowanie maksymalnej możliwej ilości plików 1kB

- Wkopiowaniu maksymalnej możliwej ilości plików 0,5kB

Wielkość plików podana w zaokrągleniu, jednak suma rozmiarów pliku najmniejszego i średniego jest dokładnie równa wielkości pliku największego.

Spodziewany rezultat końcowy wykonania programu:

- Wolne miejsce po ostatniej operacji dokładnie równe wolnemu miejscu po pierwszym wkopiowywaniu.
- Kolejność plików w pamięci plikowej:
    - 1kB
    - 0,5kB
    - 1,5kB
    - 1kB
    - 0,5kB
    - 1,5kB
    - ...

Następnie kopiuję z dysku wirtualnego pliki o numerach 10, 7 i 3, które powinny być odpowiednio plikami o rozmiarach 0,5kB, 1kB i 1,5kB. Ostatecznie porównuję je z plikami pierwotnymi za pomocą polecenia cmp.

Całość dokonuję wywołaniem ./1skrypt.sh > result poprzez skrypt:

home > adam > minix > minix_usr > FS > 1skrypt.sh

```
1  #!/bin/sh
2  rm testDisk1
3  rm wynik1
4  rm wynik2
5  rm wynik3
6  ./a.out testDisk1 10000
7  cmp wynik1 test1
8  cmp wynik2 test2
9  cmp wynik3 test3
10 echo "END"
```

Zawartość pliku result (po delikatniej zmianie formatowania dla zwiększenia czytelności wyników:

home > adam > minix > minix_usr > FS > result

```
Wkopiowywanie plikow 1,5kB
write_to_vfile(): Not enough space!

Free space: 6
Disk space: 10000

name: test0, size: 1549, begin: 700, end: 2249
name: test1, size: 1549, begin: 2249, end: 3798
name: test2, size: 1549, begin: 3798, end: 5347
name: test3, size: 1549, begin: 5347, end: 6896
name: test4, size: 1549, begin: 6896, end: 8445
name: test5, size: 1549, begin: 8445, end: 9994

Usuwanie codrugiego pliku 1,5kB
No such file!

Free space: 4653
Disk space: 10000

name: test1, size: 1549, begin: 2249, end: 3798
name: test3, size: 1549, begin: 5347, end: 6896
name: test5, size: 1549, begin: 8445, end: 9994

Wkopiowywanie plikow 1kB
write_to_vfile(): Not enough space!

Free space: 1629
Disk space: 10000

name: test6, size: 1008, begin: 700, end: 1708
name: test1, size: 1549, begin: 2249, end: 3798
name: test7, size: 1008, begin: 3798, end: 4806
name: test3, size: 1549, begin: 5347, end: 6896
name: test8, size: 1008, begin: 6896, end: 7904
name: test5, size: 1549, begin: 8445, end: 9994

Wkopiowywanie plikow 0,5kB
write_to_vfile(): Not enough space!

Free space: 6
Disk space: 10000

name: test6, size: 1008, begin: 700, end: 1708
name: test9, size: 541, begin: 1708, end: 2249
name: test1, size: 1549, begin: 2249, end: 3798
name: test7, size: 1008, begin: 3798, end: 4806
name: test10, size: 541, begin: 4806, end: 5347
name: test3, size: 1549, begin: 5347, end: 6896
name: test8, size: 1008, begin: 6896, end: 7904
name: test11, size: 541, begin: 7904, end: 8445
name: test5, size: 1549, begin: 8445, end: 9994

Kopiowanie pliku 0,5kB
```

Po zakończeniu programu a przed END nie zostały wyświetlone żadne komunikaty polecenia cmp, co znaczy, że pliki rzeczywiście są identyczne. Stan pamięci jest zgodny z założeniami - program działa prawidłowo.

Stare Testowanie (bez implementacji usuwania plików i fragmentacji danych)

Automatyczne testy stworzyłem dla dwóch najbardziej niebezpiecznych oraz żmudnych w testowaniu ręcznym przypadków:

1. Przepełnienie listy inode'ów
2. Próbą przekroczenia pamięci dyskowej

Oba przypadki testuję poprzez wywołanie programu z odpowiednimi argumentami testowymi uruchamiającymi go w automatycznym trybie testowym. Dla ułatwienia robię to za pomocą skryptów 1skrypt.sh oraz 2skrypt.sh.

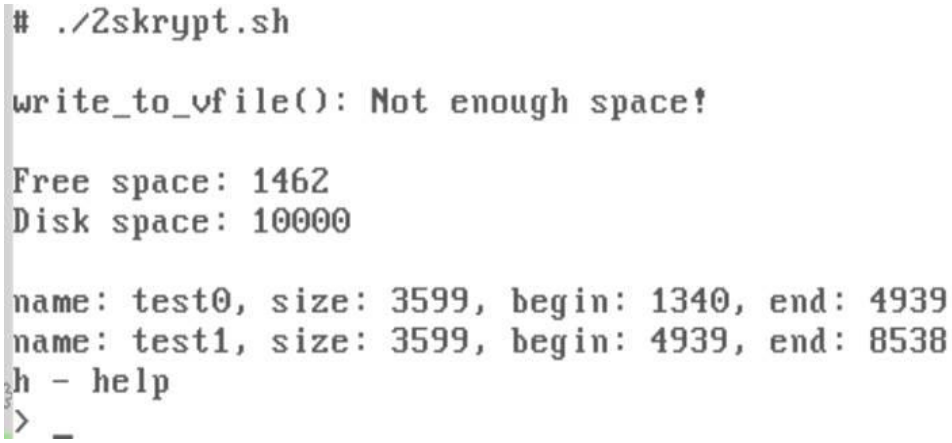
Ad. 1. Tworzony jest wirtualny dysk o zadanej (dużej) wielkości, a następnie kopiowane są na niego automatycznie (małe) pliki, tak długo aż nie zapełni się lista inode'ów. Wyświetlany jest wówczas komunikat oraz stan pamięci dysku wirtualnego.

```
# ./1skrypt.sh
copy_file(): Lack of inode's.

Free space: 8470
Disk space: 10000

name: test0, size: 19, begin: 1340, end: 1359
name: test1, size: 19, begin: 1359, end: 1378
name: test2, size: 19, begin: 1378, end: 1397
name: test3, size: 19, begin: 1397, end: 1416
name: test4, size: 19, begin: 1416, end: 1435
name: test5, size: 19, begin: 1435, end: 1454
name: test6, size: 19, begin: 1454, end: 1473
name: test7, size: 19, begin: 1473, end: 1492
name: test8, size: 19, begin: 1492, end: 1511
name: test9, size: 19, begin: 1511, end: 1530
h - help
>
```

Ad. 2. Ad. 1. Tworzony jest wirtualny dysk o zadanej (małej) wielkości, a następnie kopiowane są na niego automatycznie (duże) pliki, tak długo aż nie zapełni się lista pamięć dyskowa. Wyświetlany jest wówczas komunikat oraz stan pamięci dysku wirtualnego.



Testowanie w sposób automatyczny pozostałych sytuacji, przed którymi zabezpieczyłem program jest niesensowne i łatwiej ręcznie sprawdzić zachowanie programu podczas ich zaistnienia ze względu na ich specyfikę.