**ЛАБОРАТОРНА РОБОТА #4**

З дисципліни «Операційні системи»

На тему «Файлові системи»

|  |  |
| --- | --- |
| **Виконав**  Студент групи КВ-21  Комарницький О.Б. | **Перевірив**  Ст. викл. кафедри СПіСКС  Дробязко І.П. |

|  |  |  |
| --- | --- | --- |
| **Оцінка** | **Дата** | **Підпис** |
|  |  |  |

**Варіант #8**

**Завдання:**

Відповідно до моделі файлової системи FAT-16 створити таблицю розміщення файлів. Максимальна кількість кластерів має бути визначена попередньо. Необхідно імітувати процеси створення файлів заданого у кластерах розміру, зменшення файлів на задану кількість кластерів та знищення файлів. Передбачити пошук файлів за їх ідентифікатором та роздруківку номерів кластерів, які займає файл.

**Код програми мовою C++**

***Main.cpp***

#include "Interface.h"

#include <iostream>

#include <conio.h> // \_getch()

using namespace std;

void menu();

int main()

{

system("mode 650");

previous\_load();

while (!cin.fail())

menu();

return 0;

}

***Interface.h***

struct File

{

char \* name;

unsigned int first\_cluster;

unsigned int size;

};

struct Settings

{

unsigned int clusters\_number = 0xFFFF;

};

void previous\_load();

void add\_file(char \* name, unsigned int n);

void red\_file(char \* name, unsigned int n);

void mem\_dump(unsigned int from, unsigned int to);

void output\_fat();

***Interface.cpp***

#include "Interface.h"

#include <vector>

using namespace std;

Settings settings;

vector<int> Memory;

vector<File> FAT;

void previous\_load()

{

for (unsigned int i = 0; i <= settings.clusters\_number; ++i)

Memory.push\_back(0x0000);

// First and second can't be used

Memory[0] = 0x4449;

Memory[1] = 0x00FF;

}

void add\_file(char \* name, unsigned int n)

{

// Find out whether we have free space

unsigned int free\_clusters = settings.clusters\_number - 2;

for (auto & f : FAT)

free\_clusters -= f.size;

if (free\_clusters < n)

{

printf("No space left\n");

return;

}

File f;

f.size = n;

f.name = name;

// Search first free cluster

for (unsigned int i = 2; i <= settings.clusters\_number - 1; ++i)

{

if (Memory[i] == 0x0000)

{

f.first\_cluster = i;

break;

}

}

unsigned int tmp = f.first\_cluster;

unsigned int cur\_cluster = tmp + 1;

while (n - 1) // Last cluster for 0xFFFF

{

if (Memory[cur\_cluster] == 0x0000)

{

Memory[tmp] = cur\_cluster;

tmp = cur\_cluster;

n--;

}

cur\_cluster++;

}

Memory[tmp] = 0xFFFF;

FAT.push\_back(f);

}

void red\_file(char \* name, unsigned int n)

{

unsigned long int cluster\_number = 0x10000;

File \* file = &FAT[0];

// Find file in FAT

for (auto & f : FAT)

if (strcmp(name, f.name) == 0)

{

cluster\_number = f.first\_cluster;

file = &f;

}

if (cluster\_number == 0x10000 || n >= file->size)

{

printf("File doesn't exist or wrong size \n", name);

return;

}

// Go to new terminate cluster of this file

int steps = n - 1;

while (steps-- > 0)

cluster\_number = Memory[cluster\_number];

// Final cluster of the file

int terminate\_cluster = cluster\_number;

// Clear redundant clusters

unsigned int next;

cluster\_number = Memory[cluster\_number];

steps = file->size - n;

while (steps--)

{

next = Memory[cluster\_number];

Memory[cluster\_number] = 0x0000;

cluster\_number = next;

}

Memory[terminate\_cluster] = (n == 0) ? 0x0000 : 0xFFFF; // If n == 0 then clear memory

// Remake file info in FAT

if (n)

file->size = n;

else // remove file

for (auto f = FAT.begin(); f != FAT.end(); ++f)

if (strcmp(name, f->name) == 0)

{

FAT.erase(f);

return;

}

}

void mem\_dump(unsigned int from, unsigned int to) // 0..settings.clusters\_number

{

if (to > settings.clusters\_number - 0xf || from > settings.clusters\_number - 0xf || to % 0x10 != 0 || from % 0x10 != 0)

{

printf("Wrong bounds \n");

return;

}

for (unsigned int i = from; i <= to + 0x000f; ++i)

{

if (i % (0x000F + 1) == 0) // new line

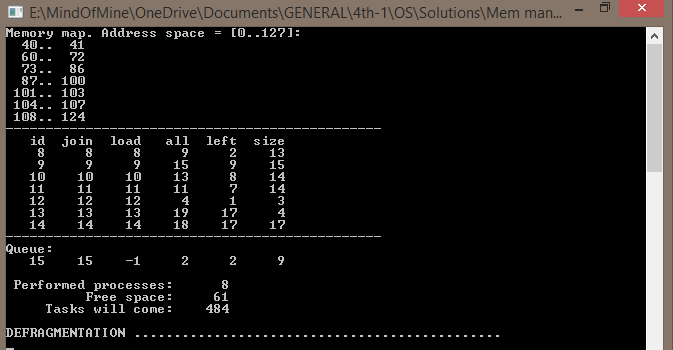
printf("\n%4x: ", i);

printf("%4x ", Memory[i]);

}

}

**Результати**

Стан пам’яті до фрагментації:

Стан пам’яті після фрагментації:

