Университет ИТМО

Кафедра прикладной математики и информатики

Системное программное обеспечение

Лабораторная работа № 2

Выполнил:

Сорокин Юрий

Группа Р3217

Преподаватель:

Зыков А.Г.

СПб

2016 г.

# Лабораторная работа № 2

#include <iostream>

#include <windows.h>

#include <fstream>

#include "shell.h"

#define SIZE 4194304

using namespace std;

int main() {

void \*memBuff = malloc(SIZE \* sizeof(int));

void \*end\_of\_memBlock = memBuff;

printf("%16s|%30s|%30s|%30s\n", "Count of numbers", "Allocation in heap by NEW", "Allocation in heap by pointer",

"File mapping");

int j = 0;

for (int i = 1024; i < SIZE; i \*= 2) {

generateFile(i, j);

int \*arrayPoint = (int \*) memBuff;

end\_of\_memBlock += i \* sizeof(int);

char \*path = (char \*) malloc(50 \* sizeof(char));

char \*pathMap = (char \*) malloc(50 \* sizeof(char));

char \*pathPointer = (char \*) malloc(50 \* sizeof(char));

sprintf(path, "%s%d%s", "./file", j, ".txt");

sprintf(pathMap, "%s%d%s", "./fileMap", j, ".txt");

sprintf(pathPointer, "%s%d%s", "./filePointer", j, ".txt");

j++;

printf("%16d|%30d|%30d|%30d\n", i, shellSortHeap(path, i), shellSortPointer(arrayPoint, pathPointer, i),

shellSortMapping(pathMap, i));

end\_of\_memBlock = memBuff;

free(path);

free(pathMap);

free(pathPointer);

}

free(end\_of\_memBlock);

free(memBuff);

return 0;

}

#include <stdlib.h>

#include <time.h>

#include <fstream>

#ifndef LAB2\_SHELL\_H

#define LAB2\_SHELL\_H

#endif //LAB2\_SHELL\_H

using namespace std;

unsigned int shellSortMapping(char \*path, int size) {

unsigned int start = clock();

HANDLE file = CreateFile(path, GENERIC\_ALL, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

HANDLE hpMapFile;

hpMapFile = CreateFileMapping(file,

NULL,

PAGE\_READWRITE,

0,

size \* sizeof(int),

NULL);

int \*array = (int \*) MapViewOfFile(hpMapFile, FILE\_MAP\_ALL\_ACCESS, 0, 0, size \* sizeof(int));

for(int j=2;j<size;j++){

int key=array[j];

int i = j-1;

while(i>0 and array[i]>key){

array[i+1]=array[i];

i-=1;

}

array[i+1]=key;

}

UnmapViewOfFile(array);

CloseHandle(hpMapFile);

CloseHandle(file);

unsigned int stop = clock();

return stop - start;

}

unsigned int shellSortPointer(int array[], char \*path, int size) {

unsigned int start = clock();

int j;

int step = size / 2;

fstream myfile(path);

int a;

int ch = 0;

while (myfile >> a) {

array[ch] = a;

ch++;

}

myfile.close();

for(int j=2;j<size;j++){

int key=array[j];

int i = j-1;

while(i>0 and array[i]>key){

array[i+1]=array[i];

i-=1;

}

array[i+1]=key;

}

fstream myfileout(path);

for (int k = 0; k < size; ++k) {

int a = array[k];

myfileout << a << " ";

}

myfileout.close();

unsigned int stop = clock();

return stop - start;

}

unsigned int shellSortHeap(char \*path, int size) {

unsigned int start = clock();

int j;

int \*array = new int[size];

int step = size / 2;

fstream myfile(path);

int a;

int ch = 0;

while (myfile >> a) {

array[ch] = a;

ch++;

}

myfile.close();

for(int j=2;j<size;j++){

int key=array[j];

int i = j-1;

while(i>0 and array[i]>key){

array[i+1]=array[i];

i-=1;

}

array[i+1]=key;

}

fstream myfileout(path);

for (int k = 0; k < size; ++k) {

int a = array[k];

myfileout << a << " ";

}

myfileout.close();

free(array);

unsigned int stop = clock();

return stop - start;

}

void generateFile(int size, int i) {

srand(1000);

FILE \*fileMap, \*file, \*filePointer;

char \*pathMap = (char \*) malloc(50 \* sizeof(char));

char \*pathPointer = (char \*) malloc(50 \* sizeof(char));

char \*path = (char \*) malloc(50 \* sizeof(char));

sprintf(pathMap, "%s%d%s", "./fileMap", i, ".txt");

sprintf(path, "%s%d%s", "./file", i, ".txt");

sprintf(pathPointer, "%s%d%s", "./filePointer", i, ".txt");

fileMap = fopen(pathMap, "w");

file = fopen(path, "w");

filePointer = fopen(pathPointer, "w");

srand(100);

for (int j = 0; j < size; ++j) {

int a = rand();

fwrite(&a, sizeof(int), 1, fileMap);

fprintf(file, "%d ", a);

fprintf(filePointer, "%d ", a);

}

free(path);

free(pathMap);

free(pathPointer);

fclose(file);

fclose(filePointer);

fclose(fileMap);

}

**Вывод**

При выполнении данной лабораторной работы, я ознакомился с разными способами ручного выделения памяти, а именно: метод кучи, метод базовых указателей и метод отображения файлов.