Университет ИТМО

Кафедра ИПМ

Системное программное обеспечение

Лабораторная работа № 2

Управление памятью

Работу выполнил:

Студент 2 курса

Группы № 2125

Назарьев Сергей

Санкт-Петербург

2014 г.

**Цель работы:**

Отсортировать 3 файла тремя способами:

* Используя стандартную кучу
* С использованием его отображения в памяти
* Использование базовых указателей в индексном файле

**Код программы:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#define MAX\_BUFFER 1024

#define SORT\_FILE\_1 "bin1"

#define SORT\_FILE\_2 "bin2"

#define SORT\_FILE\_3 "bin3"

**int** **cmp**(**const** **void** \* a, **const** **void** \* b) {

**return** (\*(**int**\*)a - \*(**int**\*)b);

}

**void** **sort\_heaps**(**char**\* filename) {

**FILE**\* file;

**long** **int** size;

**int** count;

**char**\* buffer;

HANDLE heap;

**int** a;

file = fopen(filename, "rb");

fseek(file, **0L**, SEEK\_END);

size = ftell(file);

fseek(file, **0L**, SEEK\_SET);

count = size / **sizeof**(**int**);

heap = HeapCreate(**0**, **0x01000**, **0**);

buffer = (**char**\*)HeapAlloc(heap, **0**, size);

fread(buffer, **sizeof**(**int**), count, file);

fclose(file);

qsort(buffer, count, **sizeof**(**int**), cmp);

}

**void** **sort\_mapping**(**char**\* filename) {

HANDLE hFile, hMap;

DWORD dwFileSize;

**char**\* memory;

hFile = CreateFile(filename, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, NULL, OPEN\_EXISTING, **0**, NULL);

hMap = CreateFileMapping(hFile, NULL, PAGE\_READWRITE, **0**, **0**, NULL);

dwFileSize = GetFileSize(hFile, NULL);

memory = (**char**\*)MapViewOfFile(hMap, FILE\_MAP\_WRITE, **0**, **0**, dwFileSize);

qsort(memory, dwFileSize/**sizeof**(**int**), **sizeof**(**int**), cmp);

UnmapViewOfFile(memory);

CloseHandle(hMap);

CloseHandle(hFile);

}

**void** **sort\_basepoints**(**char**\* filename) {

**int**\* it, end;

HANDLE hFile, hMap;

DWORD dwFileSize;

**char**\* memory;

**int**\* data;

**int** i, count;

hFile = CreateFile(filename, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, NULL, OPEN\_EXISTING, **0**, NULL);

hMap = CreateFileMapping(hFile, NULL, PAGE\_READWRITE, **0**, **0**, NULL);

dwFileSize = GetFileSize(hFile, NULL);

count = dwFileSize / **sizeof**(**int**);

memory = (**char**\*)MapViewOfFile(hMap, FILE\_MAP\_WRITE, **0**, **0**, dwFileSize);

data = (**int**\*)malloc(dwFileSize);

i = **0**;

it = (**int**\*) memory;

end = it + dwFileSize/**sizeof**(**int**);

**for**(it; it < end; it++)

data[i++] = \*it;

qsort(data, count, **sizeof**(**int**), cmp);

i = **0**;

it = (**int**\*)memory;

end = it + count;

**for**(it; it < end; it++)

(\*it) = data[i++];

UnmapViewOfFile(memory);

CloseHandle(hMap);

CloseHandle(hFile);

}

**int** **main**() {

SYSTEMTIME st, ft;

GetSystemTime(&st);

sort\_heaps(SORT\_FILE\_1);

GetSystemTime(&ft);

printf("Heaps: %d ms**\n**", (ft.wMilliseconds - st.wMilliseconds));

GetSystemTime(&st);

sort\_mapping(SORT\_FILE\_2);

GetSystemTime(&ft);

printf("Mapping: %d ms**\n**", (ft.wMilliseconds - st.wMilliseconds));

GetSystemTime(&st);

sort\_basepoints(SORT\_FILE\_3);

GetSystemTime(&ft);

printf("Basepoints: %d ms**\n**", (ft.wMilliseconds - st.wMilliseconds));

**return** **0**;

}

**Результат работы програмы:**

Heaps: 156 ms

Mapping: 60 ms

Basepoints: 67 ms

**Вывод:**

В ходе выполнения лабораторной работы были изучены основные методы управления памятью. Результаты замера времени по каждому методу дали понять, что по скорости работы работа с windows heap’ом – самая медленная, а два остальных метода примерно равны.

Наиболее простым в реализации и понимании программистом является метод распределения памяти с помощью куч.

Наиболее сложным, но практичным является метод базовых элементов, который по своей сути является модернизацией метода отображения файлов, но без необходимости каждый раз создавать новое отображение при последующем обращении к этому файлу.