САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Кафедра информатики и прикладной математики

Лабораторная работа №3

«Управление памятью Windows»

Выполнил:
студент II курса группы 2125
Припадчев Артём

Санкт-Петербург
2014

Куча — это регион зарезервированного адресного пространства. Первоначально большей его части физическая память не передается. По мере того, как программа занимает эту область под данные, специальный диспетчер, управляющий кучами (heap manager), постранично передаст ей физическую память (из страничного файла). А при освобождении блоков в куче диспетчер возвращает системе соответствующие стра ницы физической памяти.

Отображение файла в память (на память) — это такой способ работы с файлами в некоторых операционных системах, при котором всему файлу или некоторой непрерывной части этого файла ставится в соответствие определённый участок памяти (диапазон адресов оперативной памяти). При этом чтение данных из этих адресов фактически приводит к чтению данных из отображенного файла, а запись данных по этим адресам приводит к записи этих данных в файл. Примечательно то, что отображать на память часто можно не только обычные файлы, но и файлы устройств.

В реализации с использованием **индексного файла** строятся два представления файла. Одно из них представляет исходный файл, а второе – файл, содержащий отсортированные ключи. Второй файл является **индексным файлом**, каждая из записей которого содержит ключ и указатель, относящийся к исходному файлу.

Исходные коды

```
namespace SortHeap
{
    class Program
    {
        public static void Main()
        {
            IntPtr _hpHandle;
            var wordsize = IntPtr.Size;
            var datSize = Marshal.SizeOf(typeof(TestData));

            _hpHandle = Heap.HeapCreate(0, UIntPtr.Zero, UIntPtr.Zero);
            var ptr = Heap.HeapAlloc(_hpHandle, 0, (UIntPtr)datSize);
            var dataOrig = new TestData { data1 = 1, data2 = 2};
            Marshal.StructureToPtr(dataOrig, ptr, true);

            var tryd1 = Marshal.ReadInt32(ptr);
            Console.WriteLine(tryd1.ToString());

            var tryd2 = Marshal.ReadInt32(ptr + 4);
            Console.WriteLine(tryd2.ToString());

            var dataCopy = Marshal.PtrToStructure(ptr, typeof(TestData));
            Heap.HeapFree(_hpHandle, 0, ptr);
            Heap.HeapDestroy(_hpHandle);
            Console.ReadLine();
        }
    }
    struct TestData
    {
        public int data1;
        public int data2;
        public override string ToString()
        {
            return data1.ToString() + " " + data2.ToString();
        }
    }
    public class Heap
    {
        [DllImport("kernel32.dll", SetLastError = true)]
        public static extern IntPtr HeapCreate(uint flOptions, UIntPtr dwInitialsize, UIntPtr
dwMaximumSize);
```

```
        [DllImport("kernel32.dll", SetLastError = true)]
        public static extern IntPtr HeapAlloc(IntPtr hHeap, uint dwFlags, UIntPtr dwSize);

        [DllImport("kernel32.dll", SetLastError = true)]
        public static extern bool HeapFree(IntPtr hHeap, uint dwFlags, IntPtr lpMem);

        [DllImport("kernel32.dll", SetLastError = true)]
        public static extern bool HeapDestroy(IntPtr hHeap);
    }
}

namespace MappingFile
{
    class Program
    {
        static string LargeString = "A-";
        static List<int> arr = new List<int>();
        static void Main(string[] args)
        {
            CreateMemoryMappedFile("File.txt");
            Console.ReadLine();
        }

        private static void CreateMemoryMappedFile(string fileName)
        {
            FileInfo fInfo = new FileInfo(fileName);
            long length = fInfo.Length;
            MemoryMappedFile mySequentialMMF = MemoryMappedFile.CreateFromFile(fileName,
FileMode.Open, "MySequentialMap");
            int i = 0;
            for (int j = 0; j <10; j++ )
            {
                string numb1 = String.Empty;
                string numb2 = String.Empty;
                while (true)
                {
                    try
                    {
                        MemoryMappedViewStream myMMFViewStream =
mySequentialMMF.CreateViewStream(i, 1, MemoryMappedFileAccess.ReadWrite);

                        int b = myMMFViewStream.ReadByte();
                        if (b == 13)
                        {
                            i = i + 2;
                            break;
                        }
                        if (b == -1)
                            break;
                        numb1 += (char)b;

                        i++;
                    }
                    catch (Exception ex)
                    { break; }
                }
                arr.Add(int.Parse(numb1));
                Console.WriteLine(numb1);
            }
            mySequentialMMF.Dispose();

            arr.Sort();
            string fileName2 = "File2.txt";
            byte[] data = Encoding.UTF8.GetBytes(String.Join("\r\n", arr));
            FileInfo fInfo2 = new FileInfo(fileName2);
```

```
            long length2 = fInfo2.Length;
            MemoryMappedFile mySequentialMMF2 = MemoryMappedFile.CreateFromFile(fileName2,
FileMode.Create, "MySequentialMap2",100);
            for (int k = 0; k < data.Length; k++)
            {
                MemoryMappedViewStream myMMFViewStream2 = mySequentialMMF2.CreateViewStream(k, 1,
MemoryMappedFileAccess.ReadWrite);
                myMMFViewStream2.WriteByte(data[k]);
            }
        }

namespace PointerBasic
{
    class Program
    {
        static List<int> arr = new List<int>();
        static void Main(string[] args)
        {
            CreateMemoryMappedFile("File.txt");
            Console.ReadLine();
        }

        private static void CreateMemoryMappedFile(string fileName)
        {
            FileInfo fInfo = new FileInfo(fileName);
            long length = fInfo.Length;
            MemoryMappedFile mySequentialMMF = MemoryMappedFile.CreateFromFile(fileName,
FileMode.Open, "MySequentialMap");
            int i = 0;
            for (int j = 0; j < 10; j++)
            {
                string numb1 = String.Empty;
                string numb2 = String.Empty;
                while (true)
                {
                    try
                    {
                        MemoryMappedViewStream myMMFViewStream =
mySequentialMMF.CreateViewStream(i, 1, MemoryMappedFileAccess.ReadWrite);

                        int b = myMMFViewStream.ReadByte();
                        if (b == 13)
                        {
                            i = i + 2;
                            break;
                        }
                        if (b == -1)
                            break;
                        numb1 += (char)b;

                        i++;
                    }
                    catch (Exception ex)
                    { break; }
                }
                arr.Add(int.Parse(numb1));
            }
            mySequentialMMF.Dispose();
            List<int> oldArr = new List<int>(arr);
            arr.Sort();
            string fileName2 = "File2.txt";
            string dat = String.Empty;
            foreach (var elem in arr)
            {
                dat += elem.ToString() + " " + oldArr.IndexOf(elem) + "\r\n";
```

```
            }
            byte[] data = Encoding.UTF8.GetBytes(dat);
            FileInfo fInfo2 = new FileInfo(fileName2);
            long length2 = fInfo2.Length;
            MemoryMappedFile mySequentialMMF2 = MemoryMappedFile.CreateFromFile(fileName2,
FileMode.Create, "MySequentialMap2", 100);
            for (int k = 0; k < data.Length; k++)
            {
                MemoryMappedViewStream myMMFViewStream2 = mySequentialMMF2.CreateViewStream(k, 1,
MemoryMappedFileAccess.ReadWrite);
                myMMFViewStream2.WriteByte(data[k]);
            }
        }
    }
}
```

Вывод: в ходе лабораторной работы были рассмотрены следующие темы:

- использование куч Windows
- отображения файла в память
- построение индексного файла