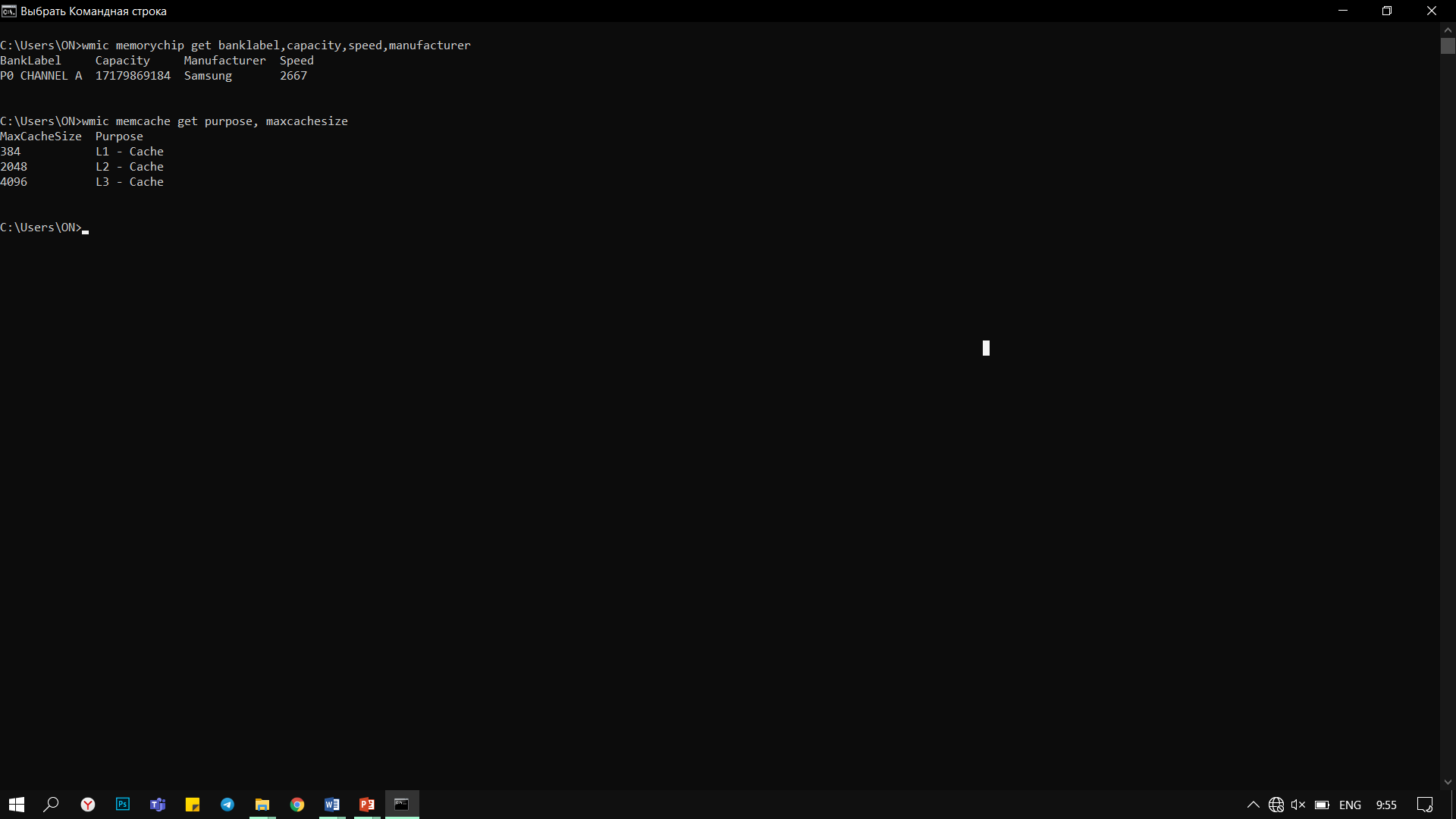
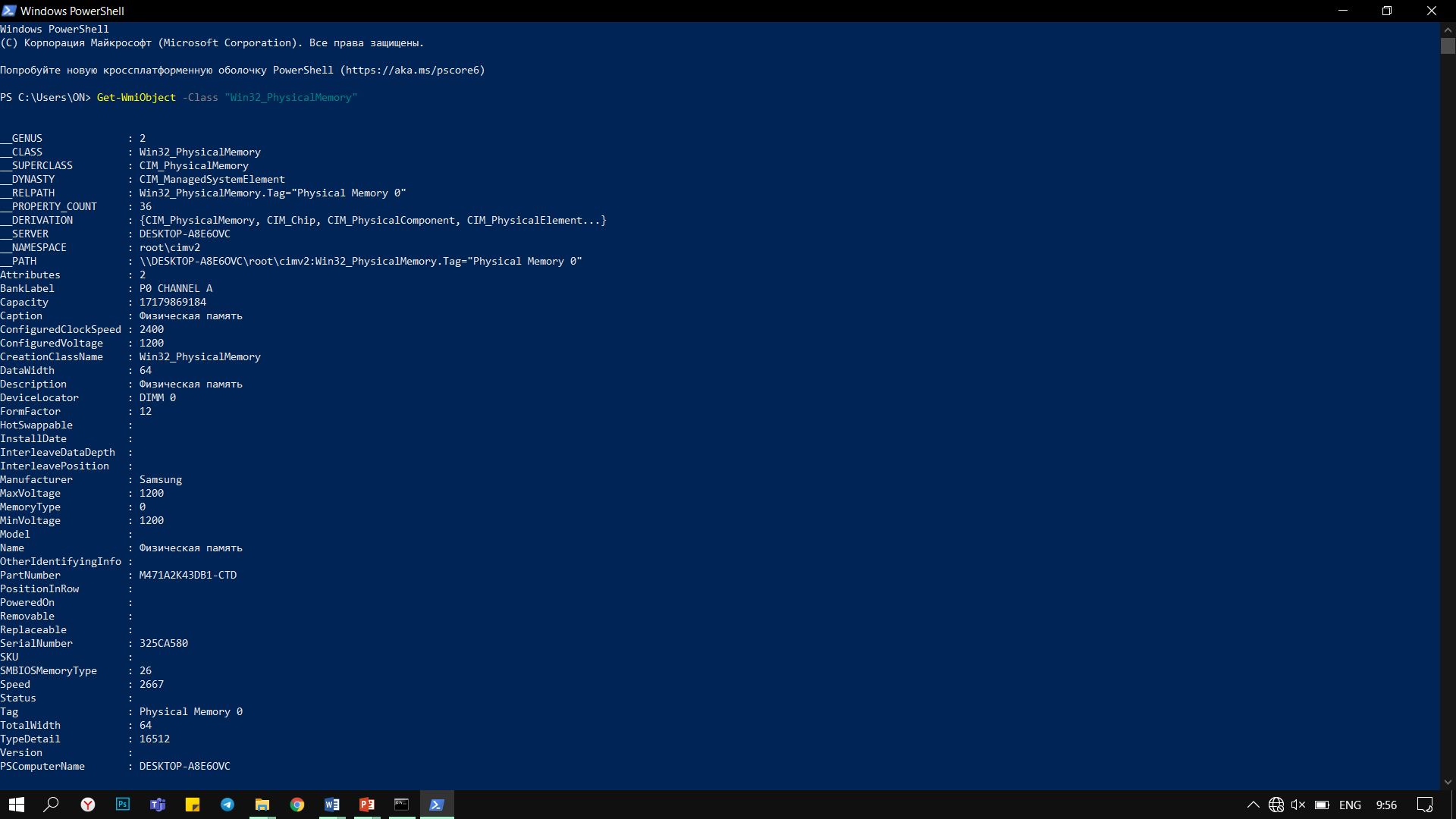
**Задание 1**

wmic memorychip get banklabel, capacity, speed, manufacturer

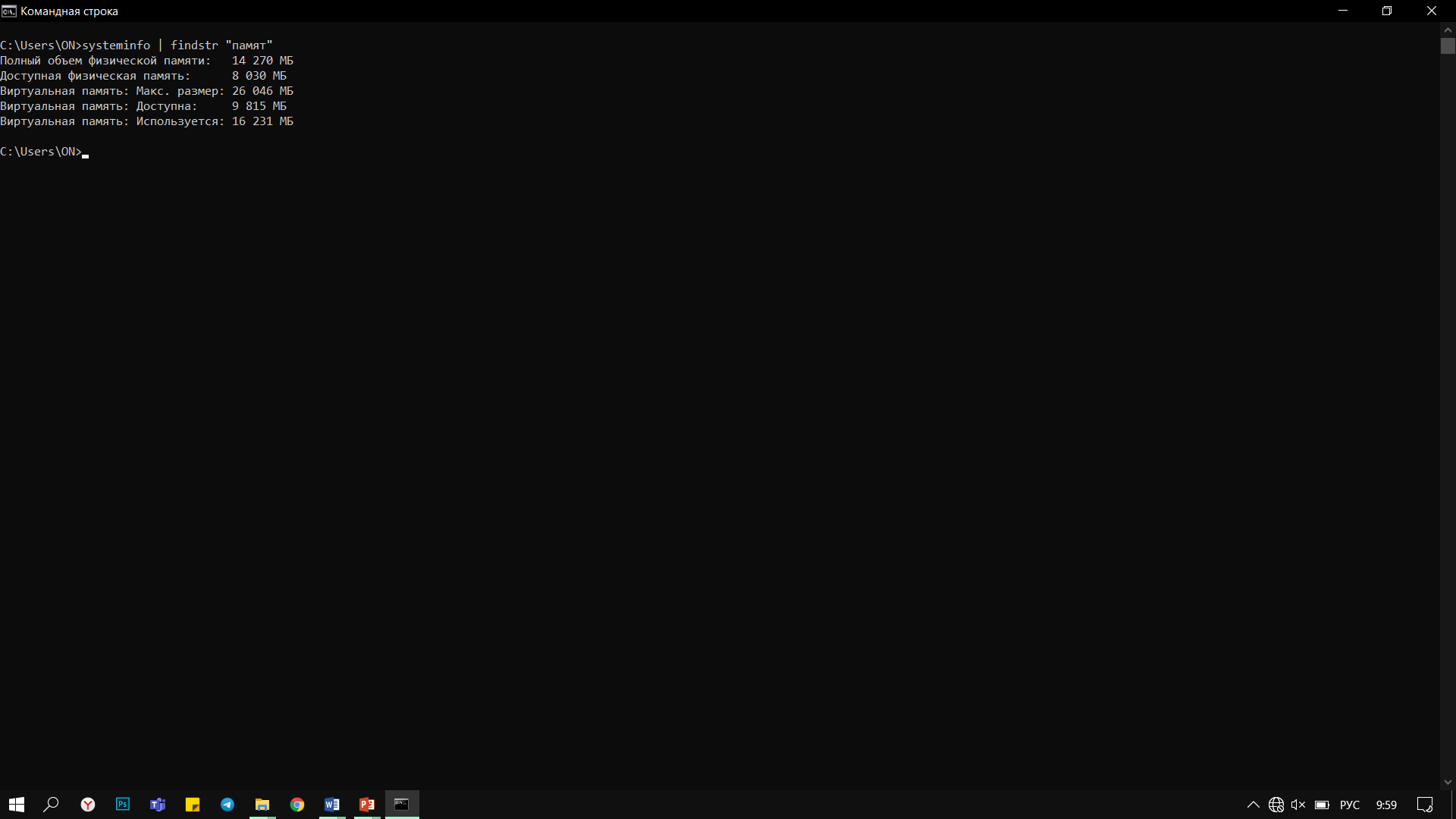
wmic memcache get purpose, maxcachesize

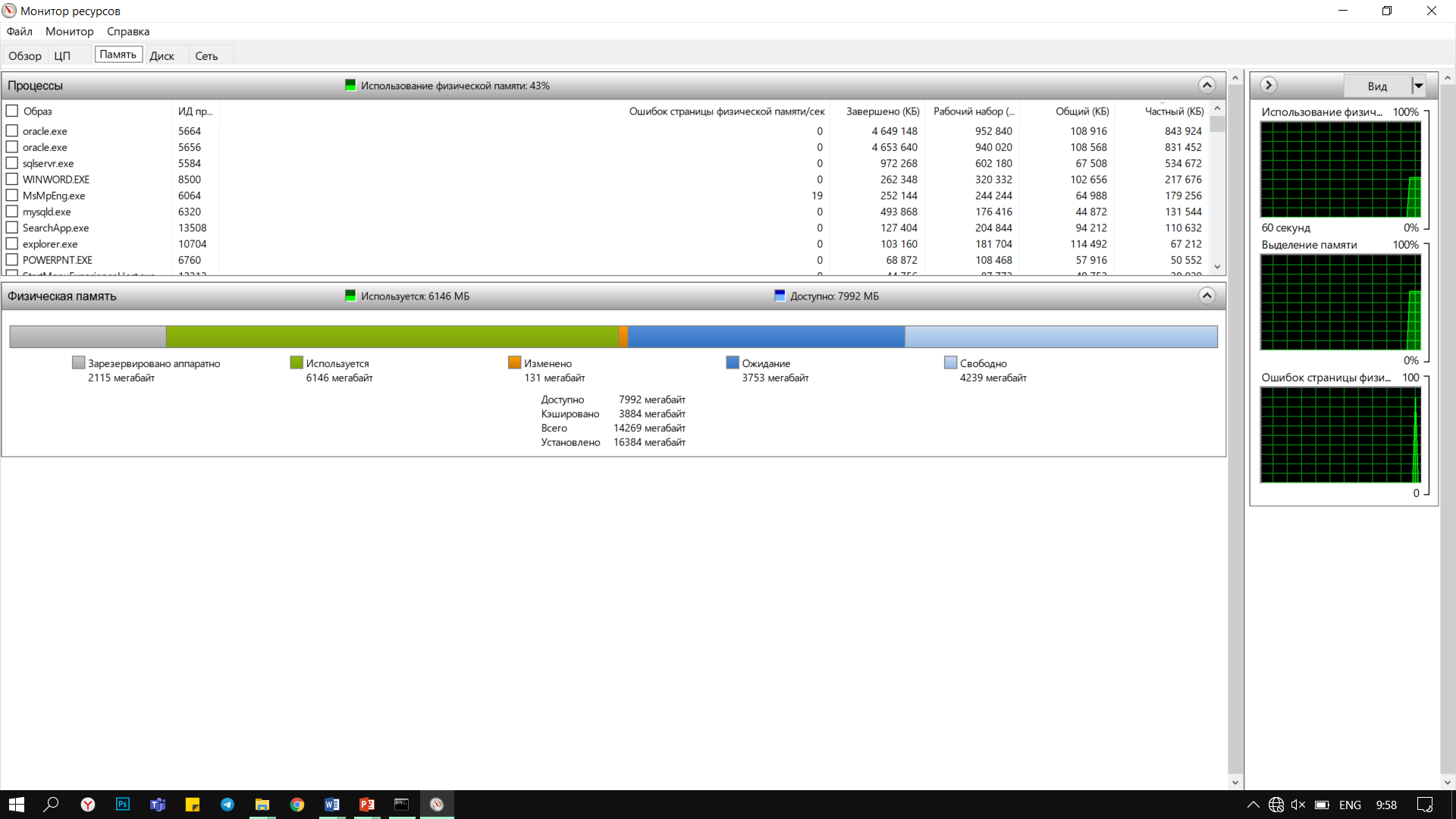


Get-WmiObject -Class "Win32\_PhysicalMemory"

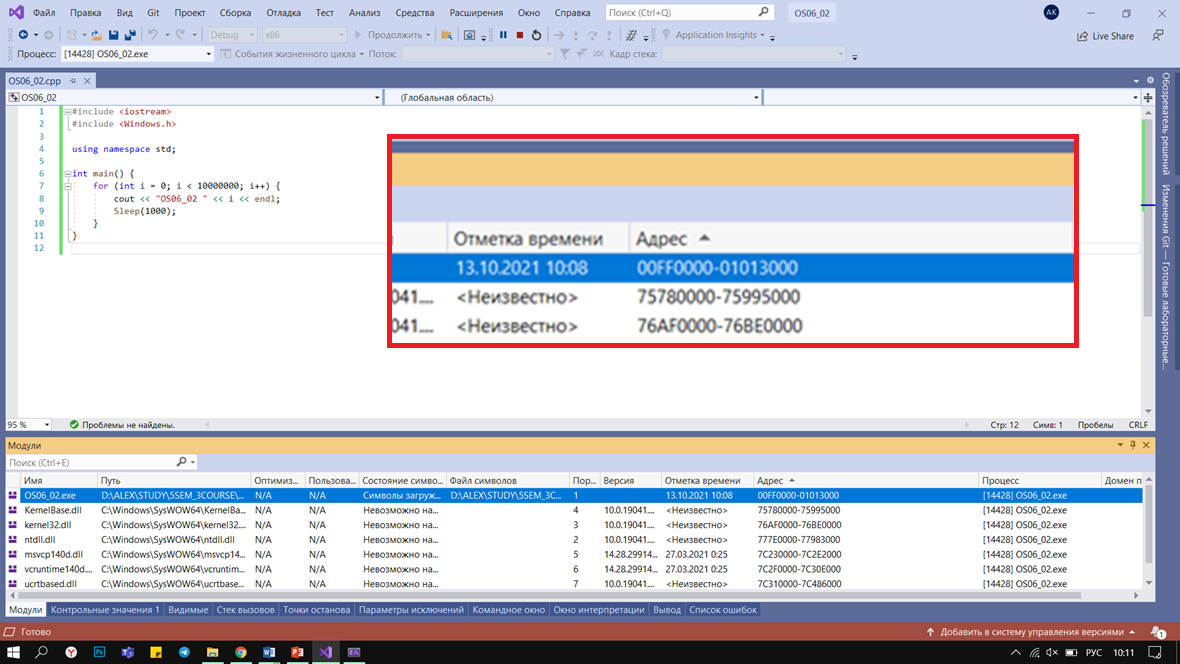


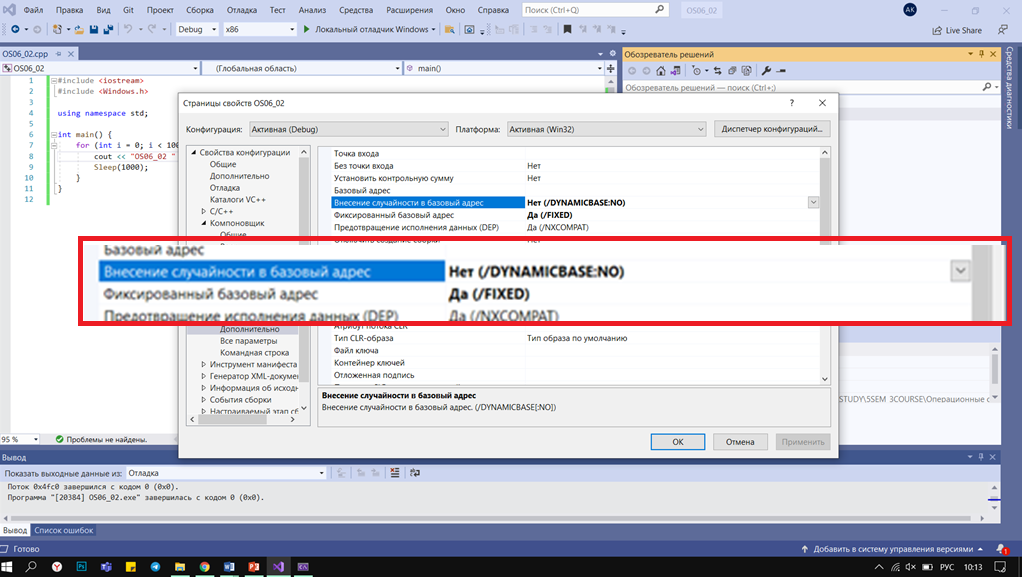
systeminfo | findstr “памят”

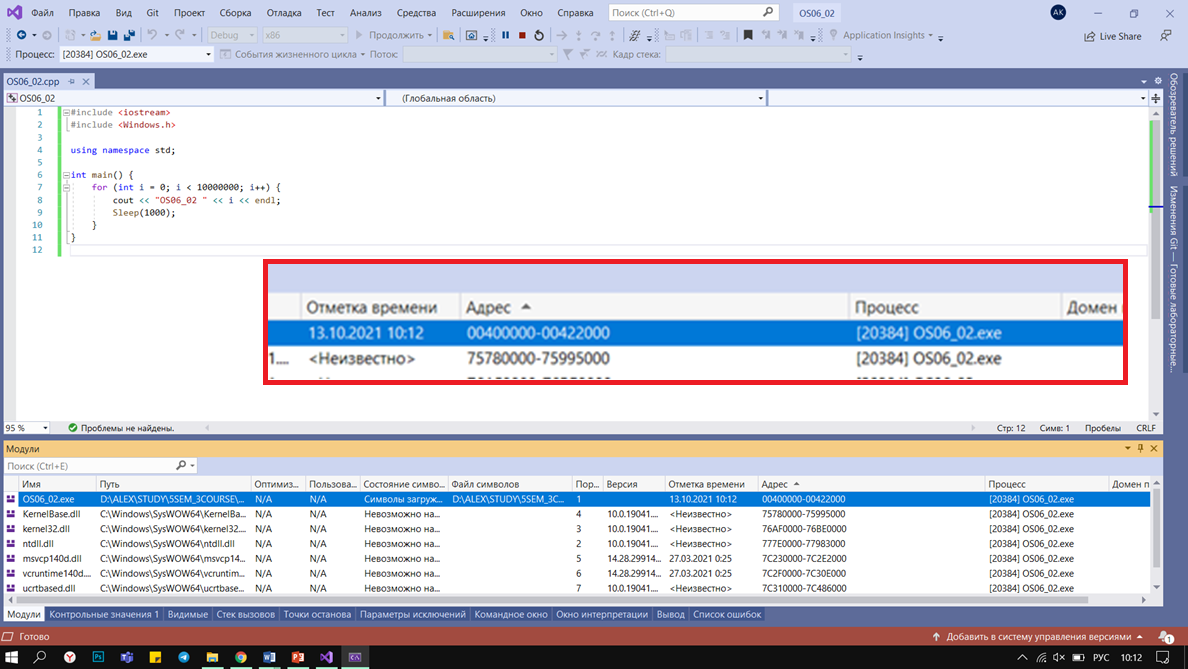




**Задание 2**

****

****

****

**Задание 3**

#include <iostream>

#include <Windows.h>

#define KB (1024)

#define PG (4 \* KB)

using namespace std;

//К CA

//о EE

//в E2

//страница CA = 202

//смещение EEE = 3822

//адрес = 202 \* PG + 3822 = 0x000CAEEE

int main() {

int pages = 256;

int countItems = pages \* PG / sizeof(int);

LPVOID xmemaddr = VirtualAlloc(NULL, pages \* PG, MEM\_COMMIT, PAGE\_READWRITE);

int\* arr = (int\*)xmemaddr;

for (int i = 0; i < countItems; i++) {

arr[i] = i;

}

cout << endl;

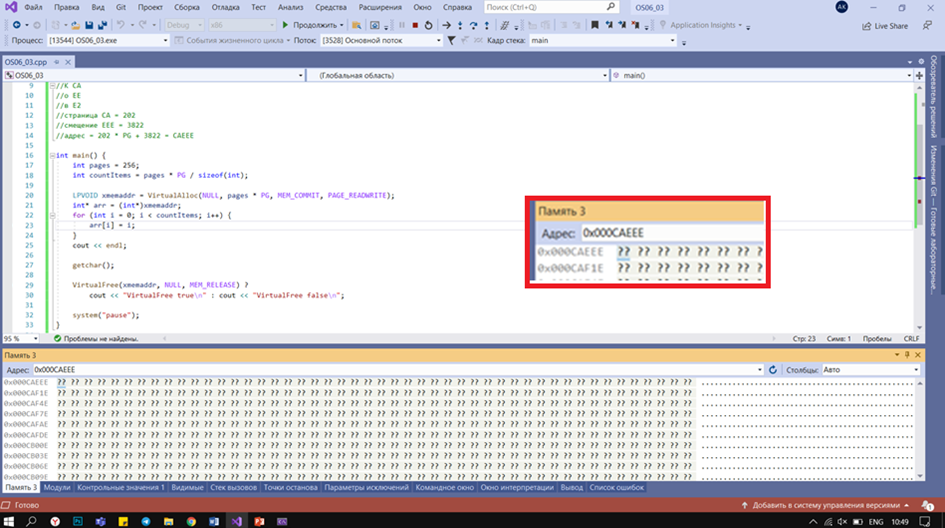
getchar();

VirtualFree(xmemaddr, NULL, MEM\_RELEASE) ?

cout << "VirtualFree true\n" : cout << "VirtualFree false\n";

system("pause");

}

****

**Задание 4**

#include <iostream>

#include <Windows.h>

using namespace std;

void sh(HANDLE pheap);

int main() {

HANDLE pheap = GetProcessHeap();

cout << "---------------Before---------------" << "\n";

sh(pheap);

{

int size = 300000;

int\* m = new int[size];

cout << "-- m = " << hex << m << ", size = " << dec << size << " \n";

}

cout << "------------After--------------\n";

sh(pheap);

}

void sh(HANDLE pheap) {

PROCESS\_HEAP\_ENTRY phe;

phe.lpData = NULL;

while (HeapWalk(pheap, &phe)) {

cout << "-- address = " << hex << phe.lpData << ", size = " << dec << phe.cbData

<< ((phe.wFlags & PROCESS\_HEAP\_REGION) ? " R" : "") // начало непрерывной области

<< ((phe.wFlags & PROCESS\_HEAP\_UNCOMMITTED\_RANGE) ? " U" : "") // нераспределенная область

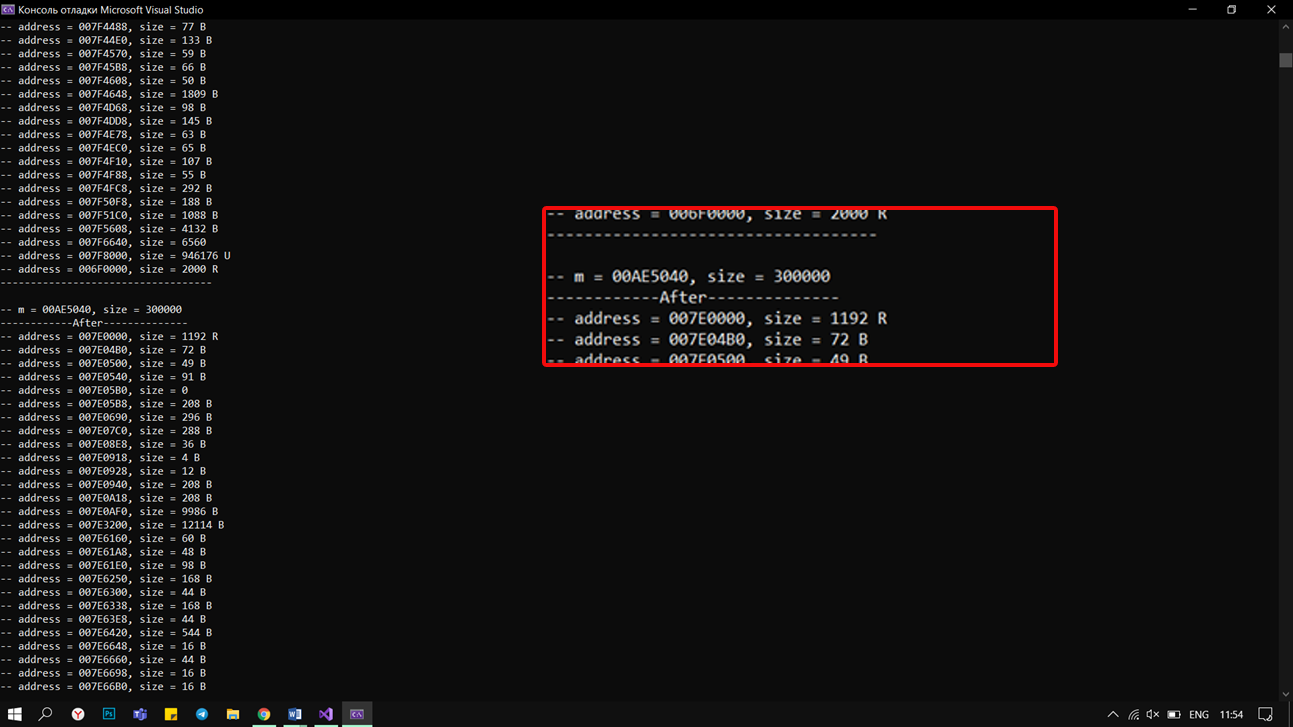
<< ((phe.wFlags & PROCESS\_HEAP\_ENTRY\_BUSY) ? " B" : "") // распределенная область

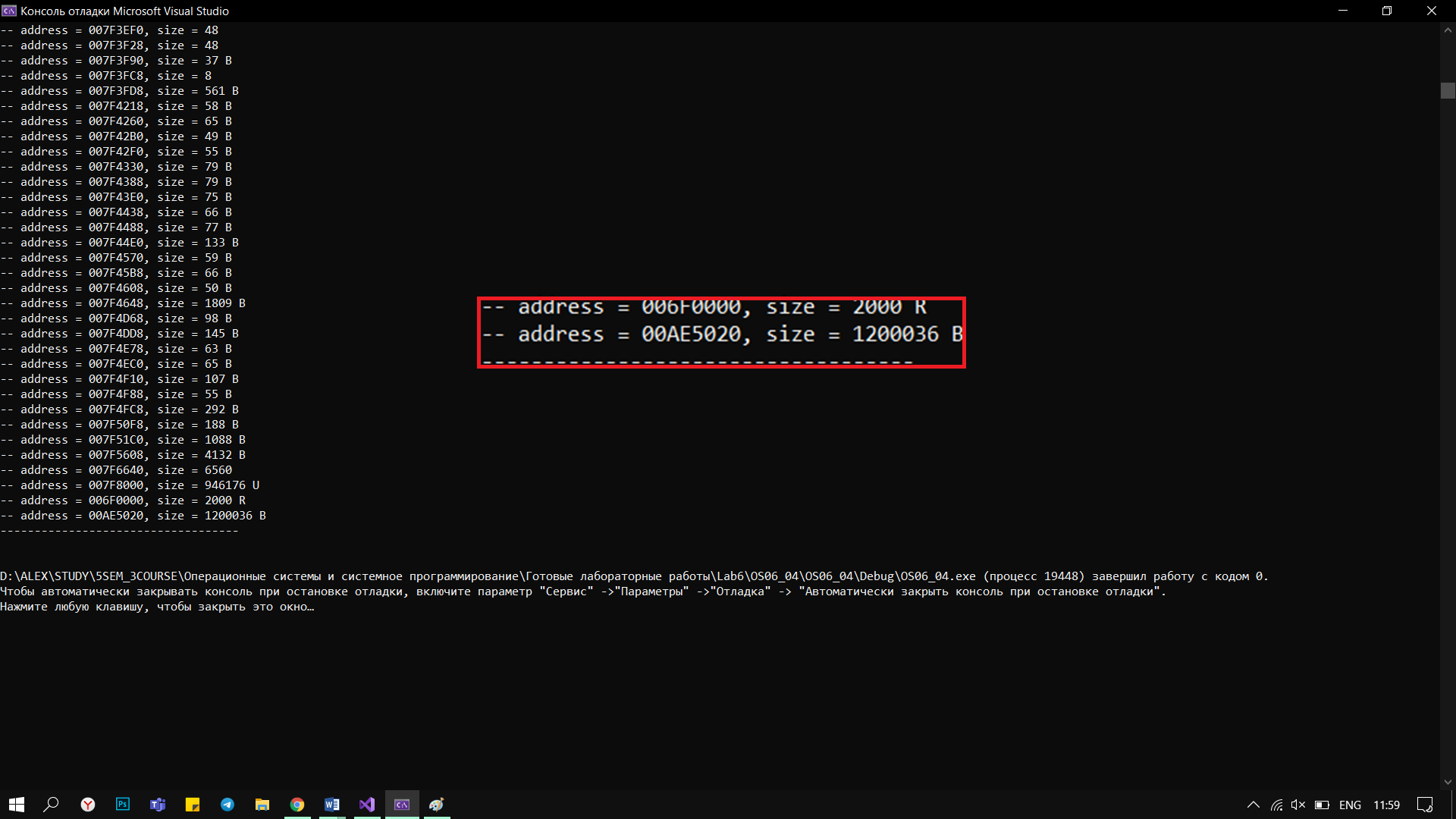
<< "\n";

}

cout << "-----------------------------------\n\n";

}

****

****

**Задание 5**

#include <iostream>

#include <Windows.h>

#define KB (1024)

using namespace std;

void sh(HANDLE pheap);

int main() {

HANDLE heap = HeapCreate(HEAP\_NO\_SERIALIZE | HEAP\_ZERO\_MEMORY, 4096 \* KB, 4096 \* 2 \* KB);

cout << "---------------Before---------------\n";

sh(heap);

int\* m = (int\*)HeapAlloc(heap, HEAP\_NO\_SERIALIZE | HEAP\_ZERO\_MEMORY, 500 \* sizeof(int));

cout << "-- m = " << hex << m << "\n";

cout << "\n------------After--------------\n";

sh(heap);

HeapFree(heap, HEAP\_NO\_SERIALIZE, m);

cout << "\n------------HeapFree--------------\n";

sh(heap);

HeapDestroy(heap);

system("pause");

}

void sh(HANDLE pheap) {

PROCESS\_HEAP\_ENTRY phe;

phe.lpData = NULL;

while (HeapWalk(pheap, &phe)) {

cout << "-- address = " << hex << phe.lpData

<< ", size = " << dec << phe.cbData

<< ((phe.wFlags & PROCESS\_HEAP\_REGION) ? " R" : "") // начало непрерывной области

<< ((phe.wFlags & PROCESS\_HEAP\_UNCOMMITTED\_RANGE) ? " U" : "") // нераспределенная область

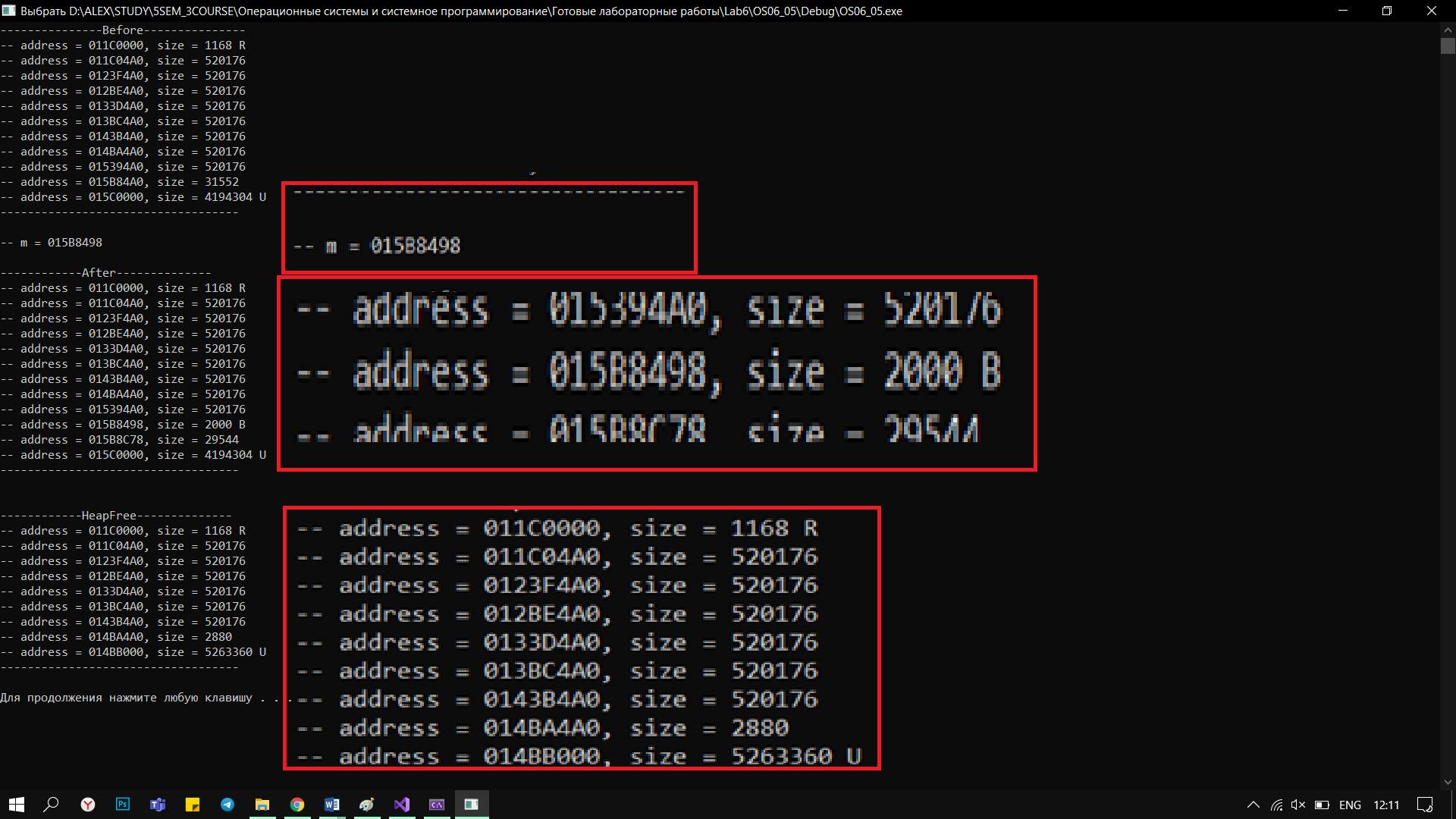
<< ((phe.wFlags & PROCESS\_HEAP\_ENTRY\_BUSY) ? " B" : "") // распределенная область

<< "\n";

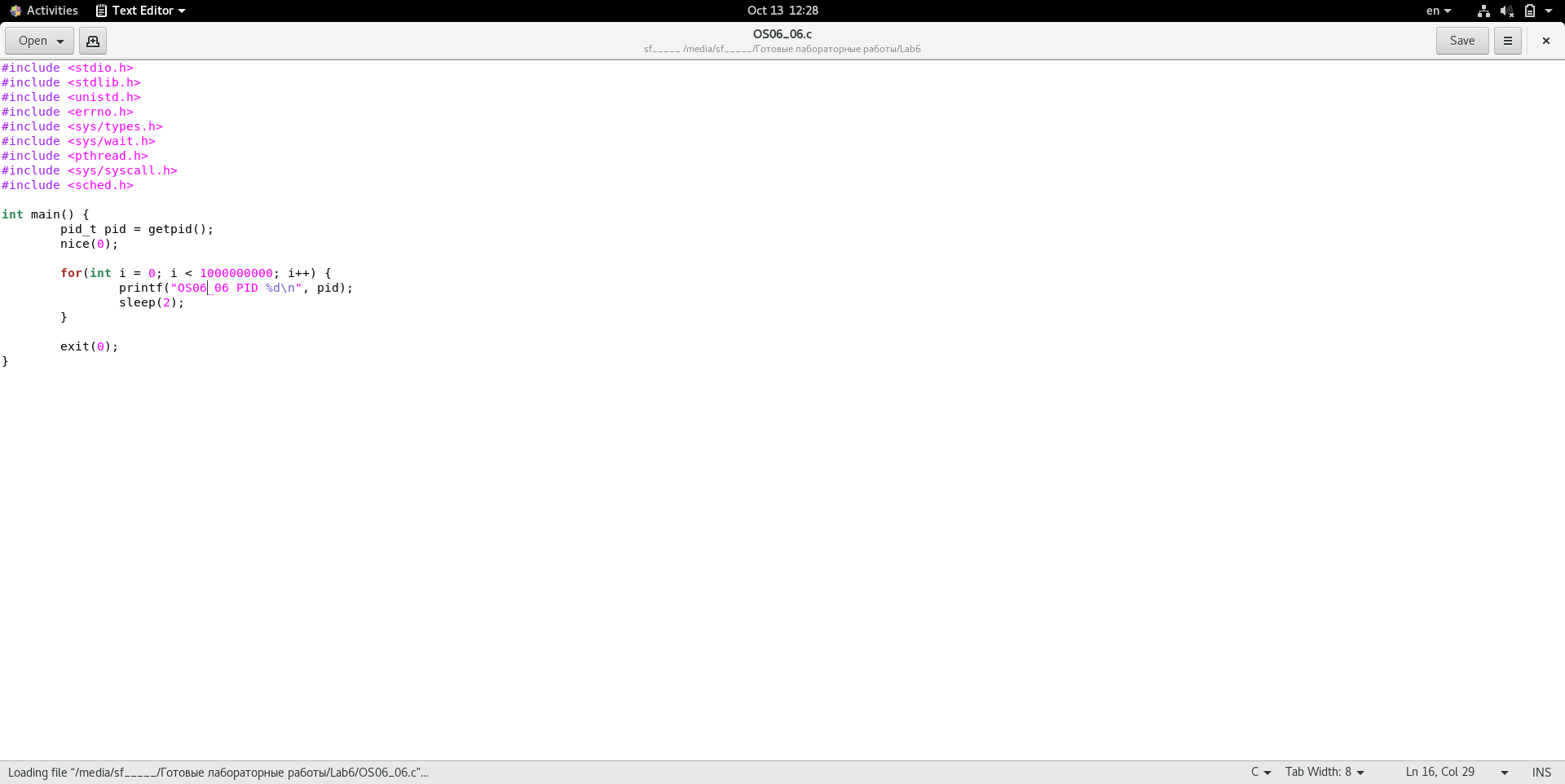
}

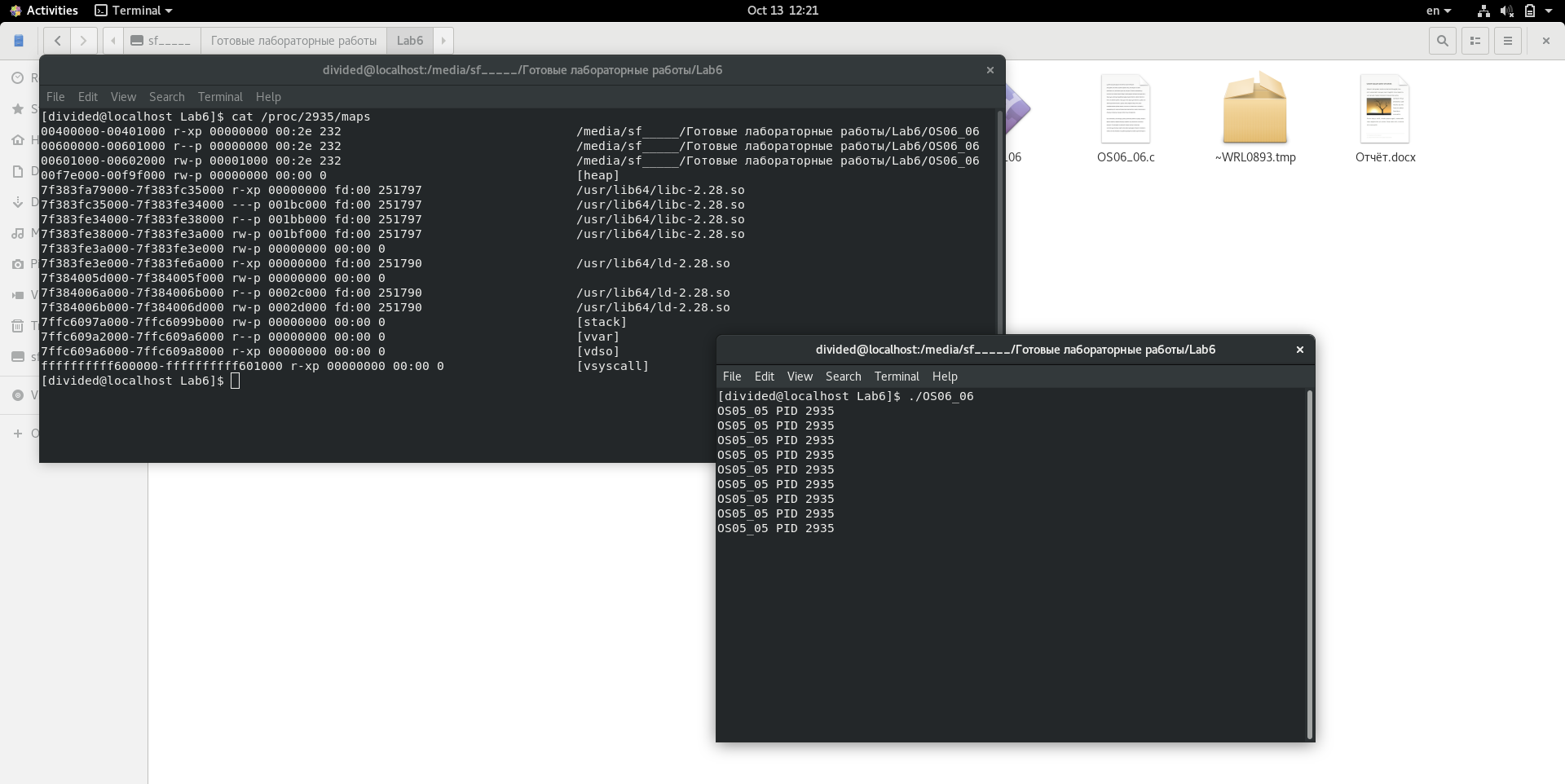
cout << "-----------------------------------\n\n";

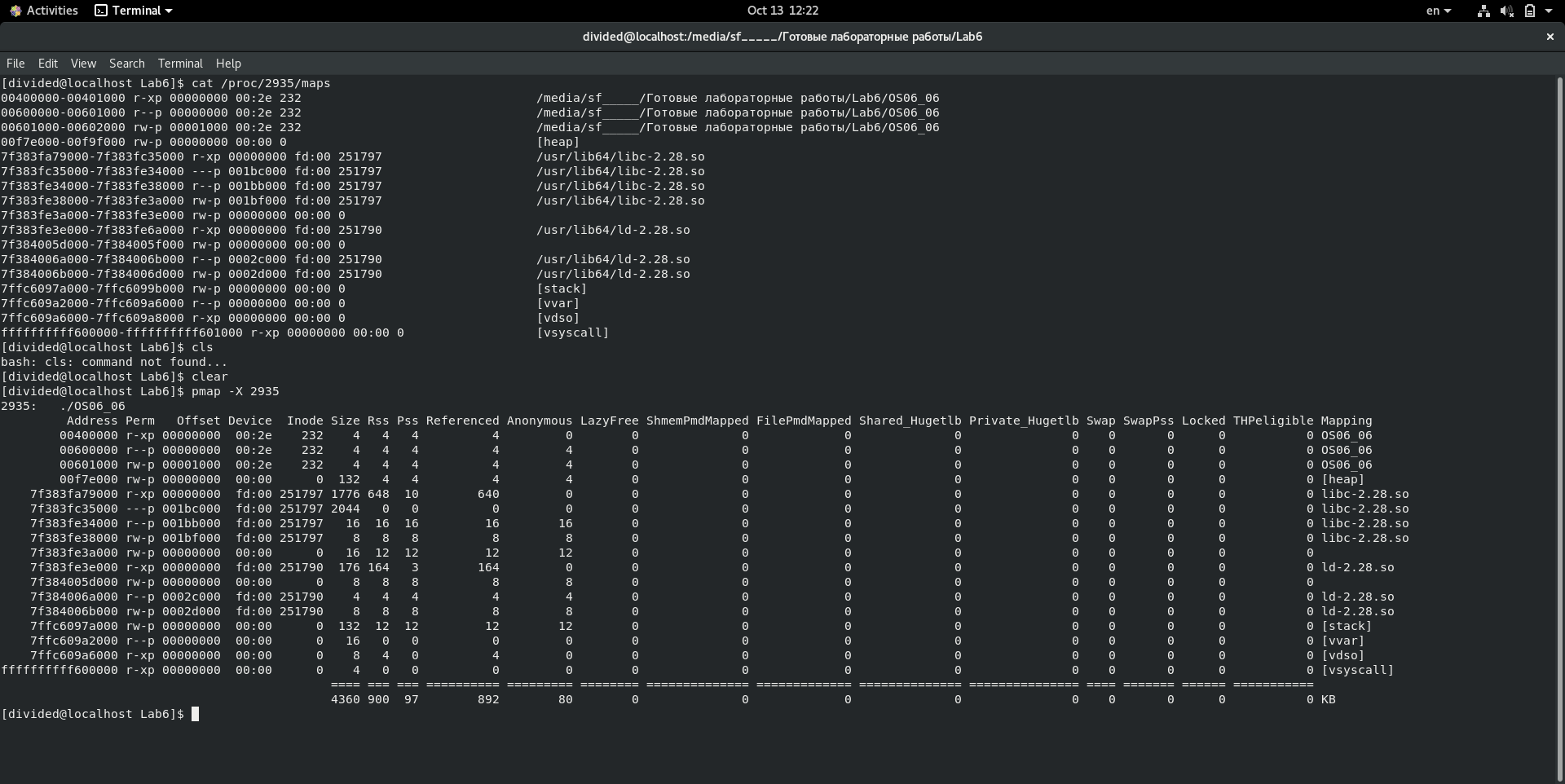
}

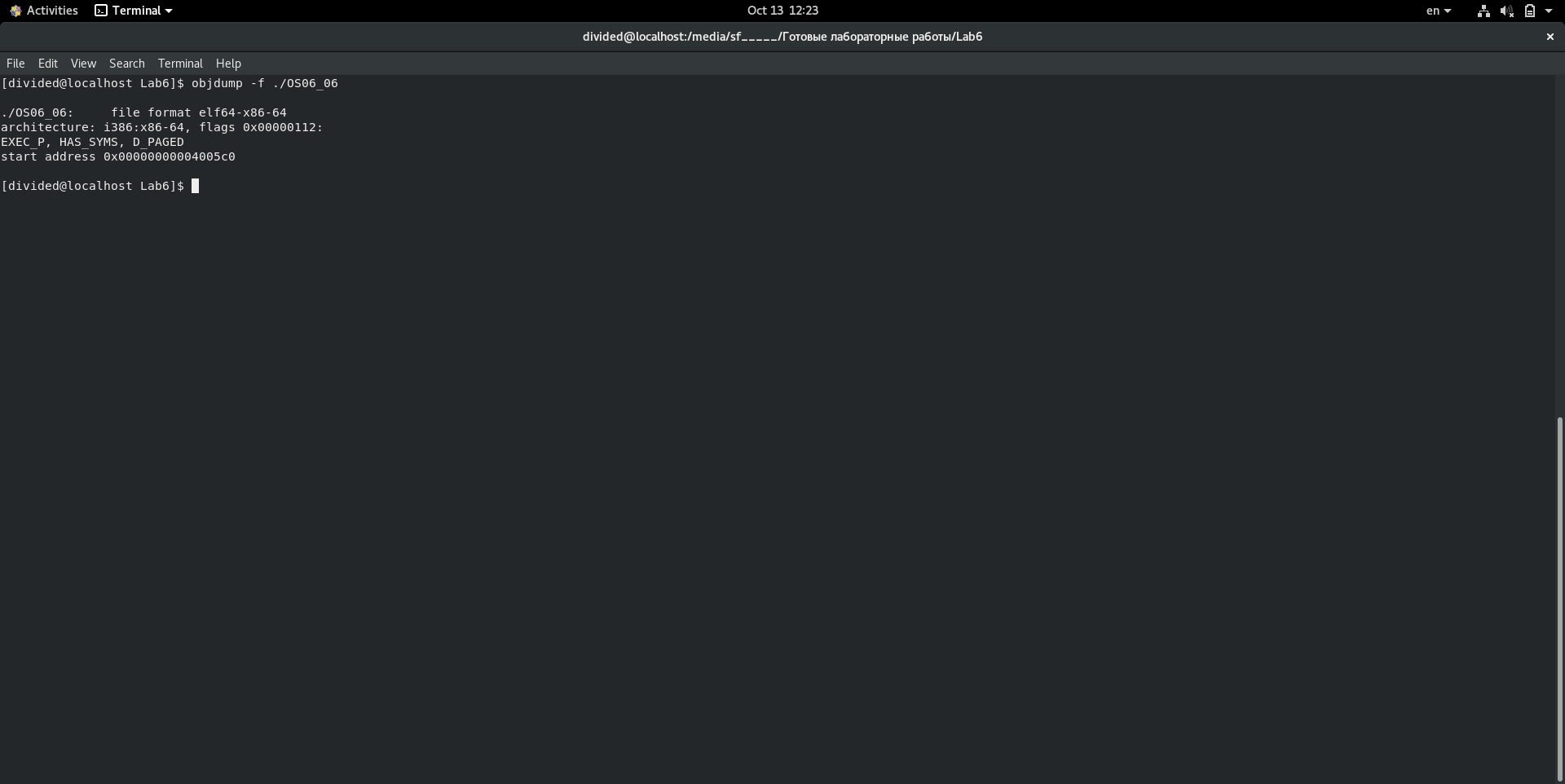
****

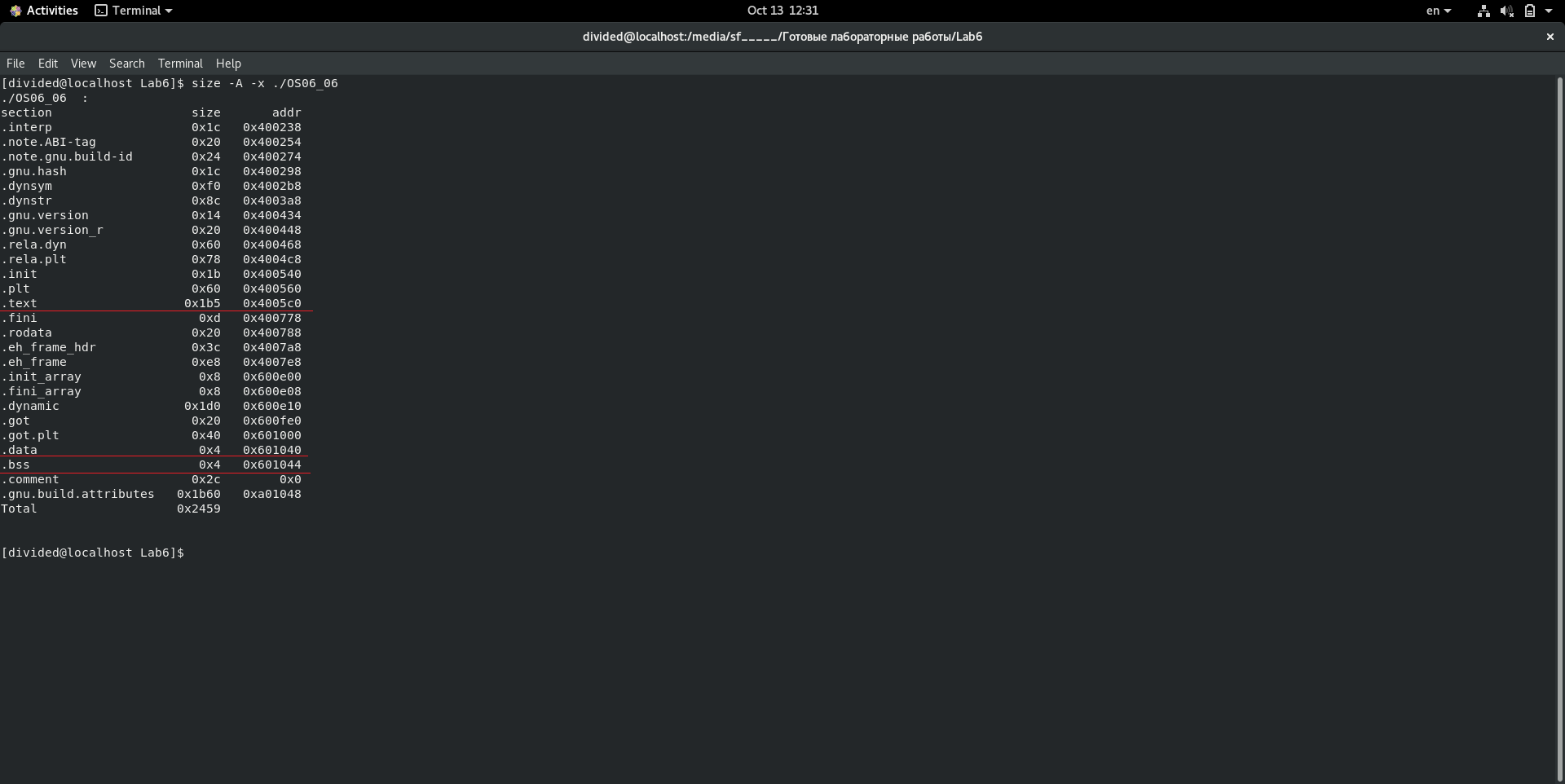
**Задание 6**

****

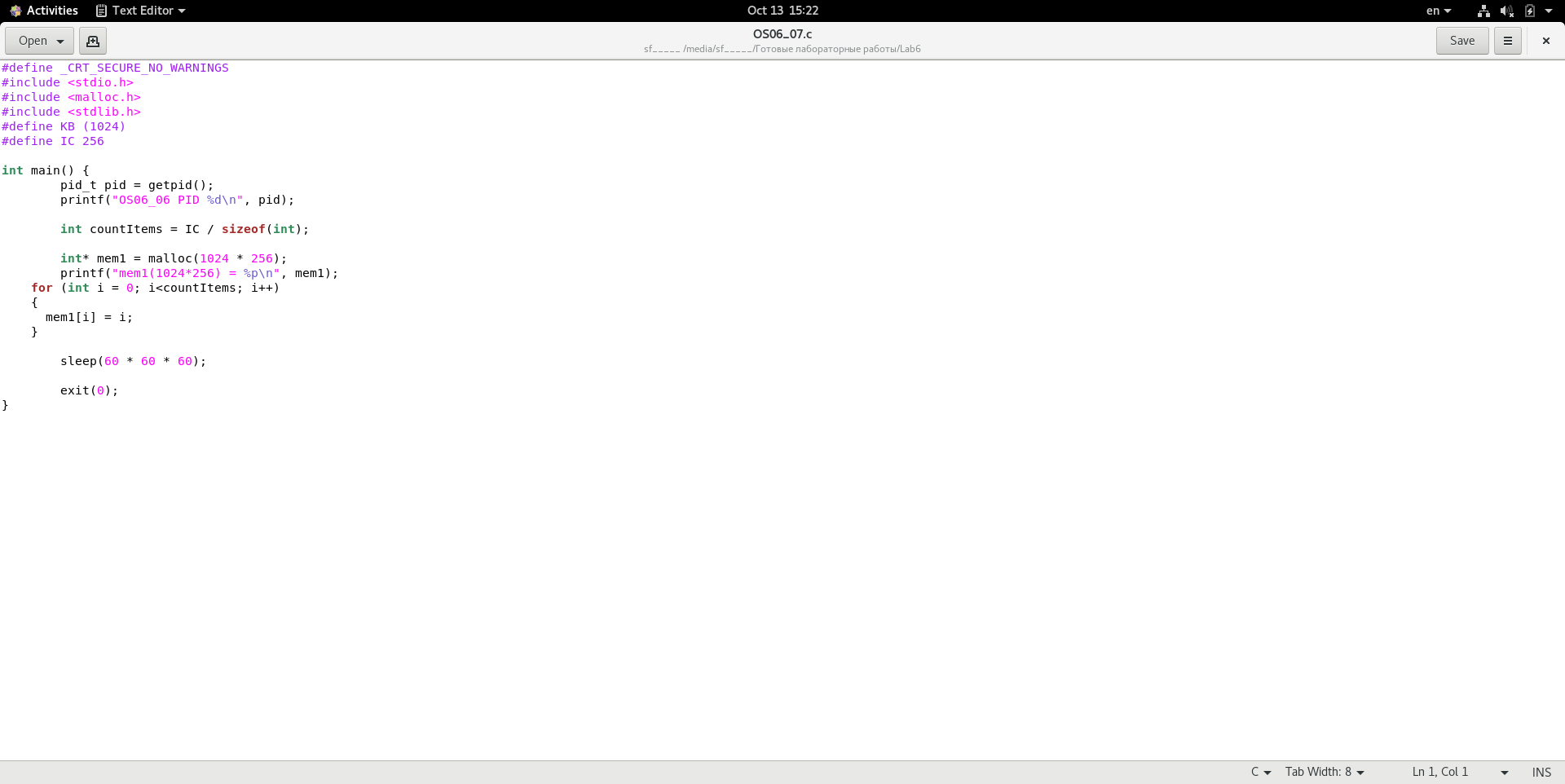
****

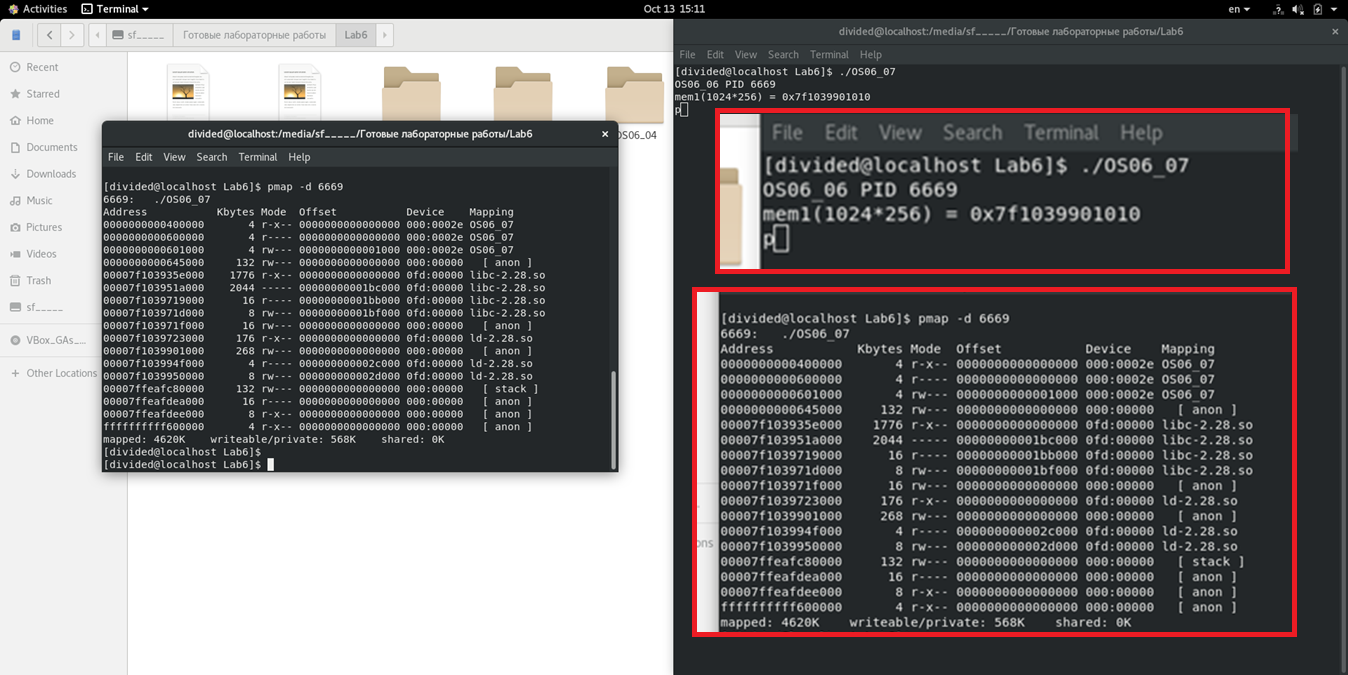
****

****

****

**Задание 7**

****

****

## Задание 1

Столбец **Capacity** (с англ. Capacity - Вместительность, Ёмкость, Объём) показывает нам объём установленных модулей памяти в байтах для каждого отдельно (в данном примере установлено два модуля памяти).

Столбик "**Speed**" отобразит скорость в МГц каждой планки.

Столбец **Manufacturer** отображает производителя модуля памяти. В зависимомти от модели и производителя модуля памяти параметр Manufacturer может быть пустым.

Команда WMIC (Windows Management Instrumentation Command) используется для получения сведений об оборудовании и системе, управления процессами и их компонентами, а также изменения настроек с использованием возможностей инструментария управления Windows (Windows Management Instrumentation или WMI).

## Задание 3

**VirtualAlloc** - выделить или зарезервировать страницы виртуальной памяти.

1. Задает начальный адрес. (**NULL** и тогда операционная система сама выберет этот адрес.)
2. Размер области в байтах
3. Тип выделения памяти: **MEM\_COMMIT** - Выполняется выделение страниц памяти для непосредственной работы с ними. Выделенные страницы заполняются нулями.
4. Защита памяти для области выделяемых страниц

**VirtualFree** - выпускает, списывает или освобождает и списывает область страниц в виртуальном адресном пространстве вызывающего процесса.

1. Указатель на базовый адрес области освобождаемых страниц.
2. Размер области памяти, подлежащей освобождению, в байтах.
3. Тип свободной операции: **MEM\_RELEASE** - Освобождает указанную область страниц. После этой операции страницы находятся в свободном состоянии.

## Задание 4

**Heap** – область памяти адресного пространства, предназначенного для использования программной фрагментов динамически выделяемой памяти (malloc, new). По умолчанию – 1MB, из них 4K сразу забирает процесс. Можно установить стартовое значение величины HEAP в параметрах Linker. По мере new(malloc) размер HEAP прирастает. Память выделяется с учетом минимизации фрагментации.

Каждый процесс имеет кучу, предоставленную операционной системой. Извлекает дескриптор кучи процесса

Ссылку на кучу процесса можно получить, воспользовавшись функцией **GetProcessHeap.**

**PROCESS\_HEAP\_ENTRY** структура - содержит информацию об элементе кучи

**HeapWalk** - Выполняет обход кучи и возвращает сведения о следующей записи.

## Задание 5

**HeapCreate** - Создает частный объект кучи, который может быть использован вызывающим процессом. Функция резервирует место в виртуальном адресном пространстве процесса и выделяет физическое хранилище для указанной начальной части этого блока.

1. Параметры выделения кучи:

**HEAP\_NO\_SERIALIZE** - Сериализованный доступ не используется, когда функции кучи обращаются к этой куче.

**HEAP\_ZERO\_MEMORY** - установить содержимое выделяемой памяти в 0

1. Начальный размер кучи, в байтах.
2. Максимальный размер кучи, в байтах.

**HeapAlloc** - Выделяет блок памяти из кучи. Выделенная память не является подвижной.

1. Дескриптор кучи, из которой будет выделена память.

# Параметры выделения кучи.

1. Количество выделяемых байтов.

## Задание 6

Каждая строка в cat /proc/$PID/maps описывает область непрерывной виртуальной памяти в процессе или потоке. Каждая строка имеет следующие поля:



* **адрес**. Это начальный и конечный адрес региона в адресном пространстве процесса
* **разрешения**. Здесь описывается доступ к страницам в регионе. Существует четыре различных разрешения: чтение, запись, выполнение и совместное использование. Если чтение /запись /выполнение отключены, вместо - /w /x. Если регион не является общим, он является приватным, поэтому p появляются вместо s. Если процесс пытается получить доступ к памяти способом, который не разрешен, генерируется ошибка сегментации. Разрешения можно изменить с помощью системного вызова mprotect.
* **смещение** - если область была сопоставлена ​​из файла (с использованием mmap), это смещение в файле где начинается отображение. Если память не была отображена из файла, это просто 0.
* **устройство**. Если регион был сопоставлен из файла, это основной и младший номер устройства (в шестнадцатеричном формате), в котором находится файл.
* **inode**. Если регион был сопоставлен с файлом, это номер файла.
* **путь**. Если регион был сопоставлен с файлом, это имя файла. Это поле пустое для анонимных сопоставленных регионов. Существуют также специальные области с именами, такими как [heap], [stack] или [vdso]. [vdso] обозначает виртуальный динамический общий объект. Он используется системными вызовами для переключения в режим ядра.

**pmap** – Использование памяти процессами.

**pmap -d** PID

Команда детально расписывает использование оперативной памяти процессами в системе.

Параметр**-x** можно использовать для предоставления информации о выделении памяти и типах отображения для каждого отображения. Для каждого сопоставления отображается объем резидентной, не совместно используемой анонимной и заблокированной памяти