Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра программной инженерии](https://www.belstu.by/fakultety/fit/vm)

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №8**

Дисциплина: Операционные системы

Тема: Управление памятью

Выполнила:

студентка 3 курса 5 группы

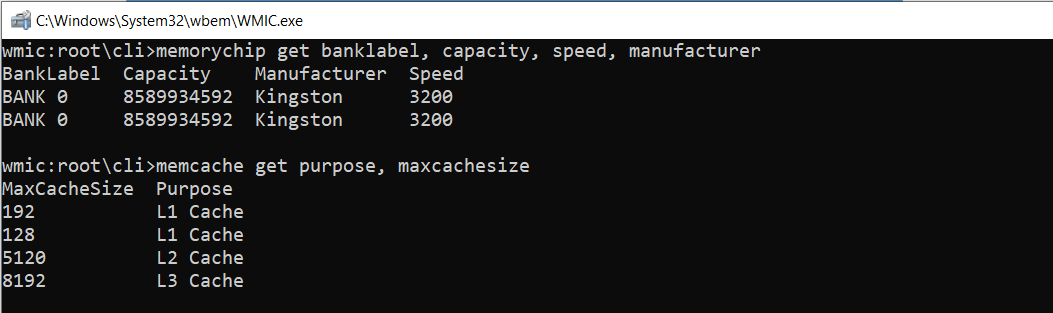
Вовна Ярослава Руслановна

Минск 2024

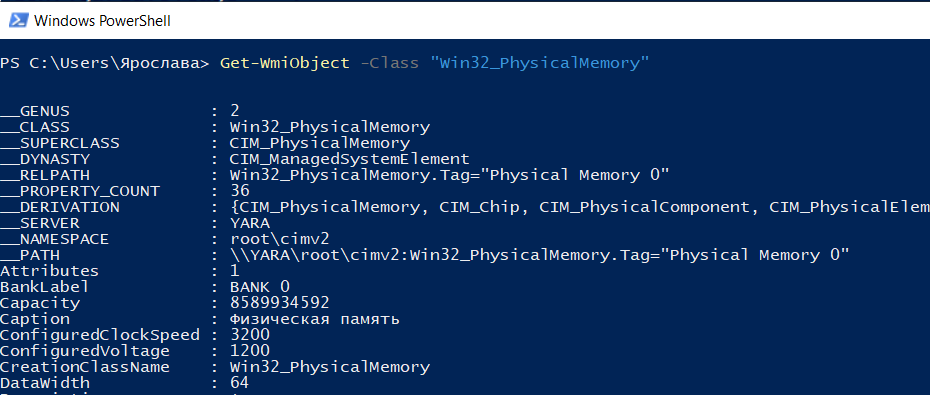
**Задание 01**

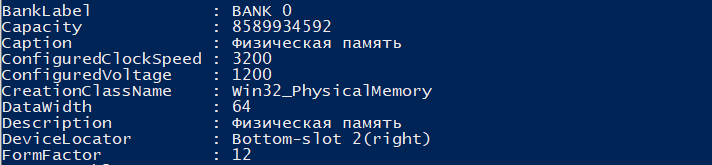
1. Информация о физической оперативной памяти компьютера через wmic

memorychip get banklabel, capacity, speed, manufacturer

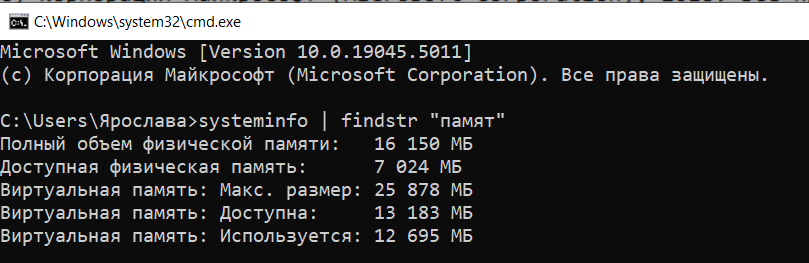


1. через powershell

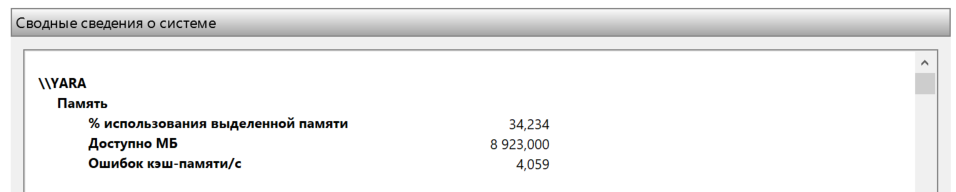
Get-WmiObject -Class "Win32\_PhysicalMemory"

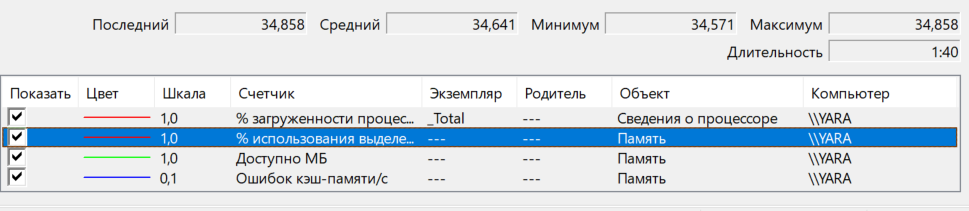


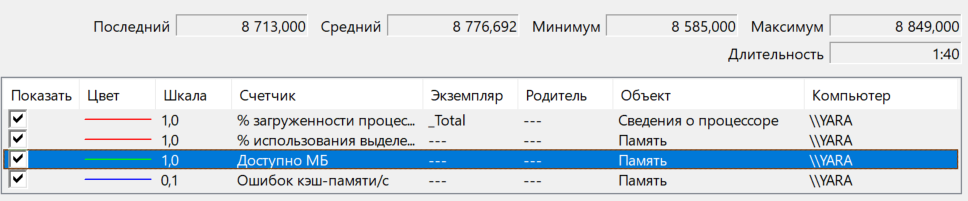
1. через systeminfo

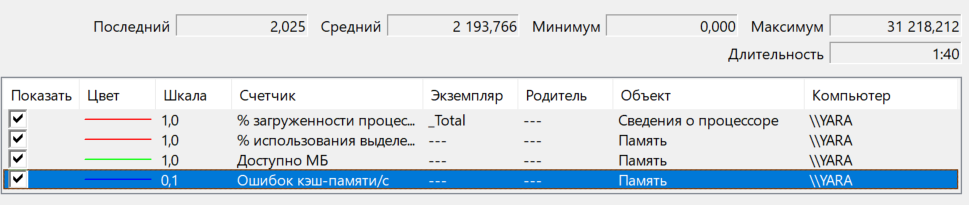


1. через perfmon









**Задание 02**

5)

#include <stdio.h>

int main()

{

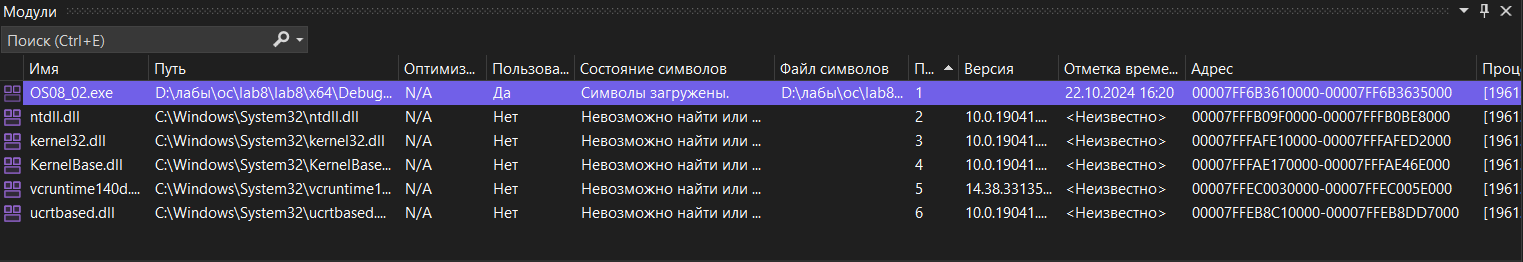
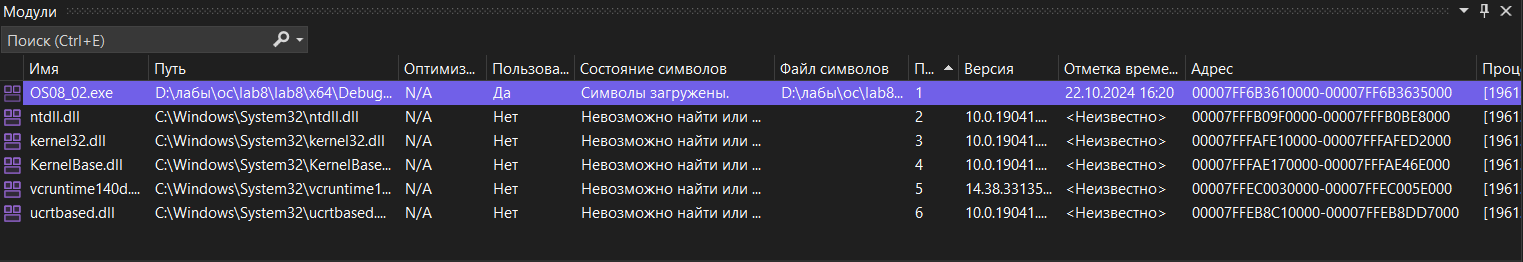
for (int i = 0; i < 1000000; i++) {

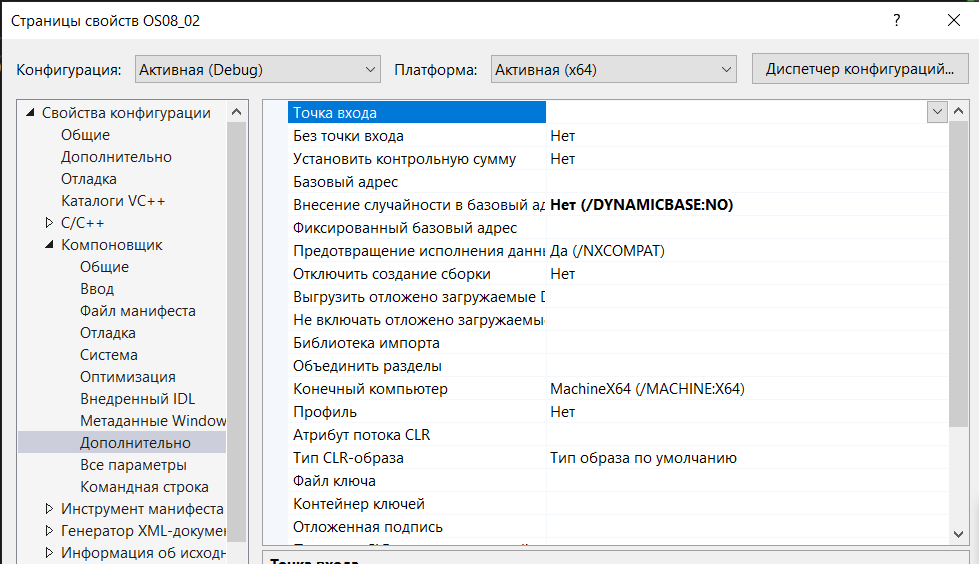
printf("Iteration: %d", i);

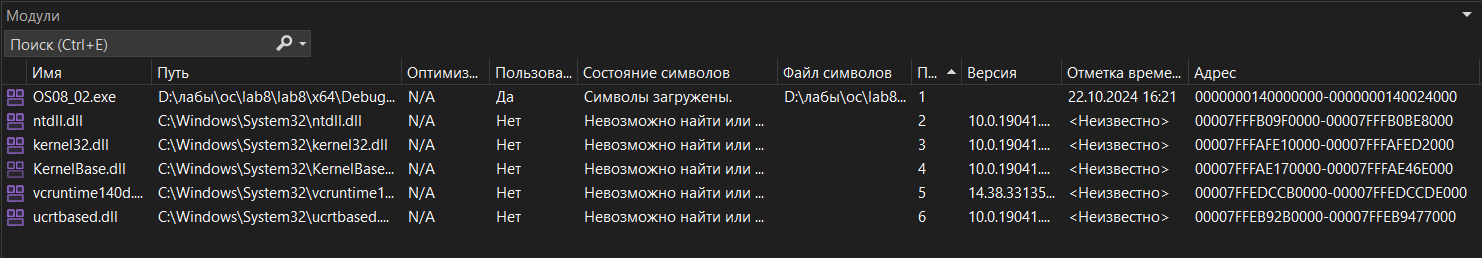
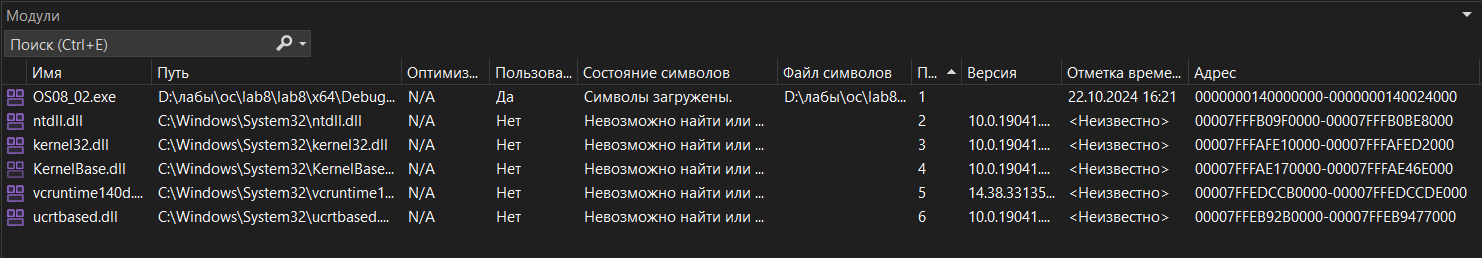
}

}

6)







**Задание 03**

7-13)

Вовна

В = C2

o = EE (1110 1110)

в = E2 (1110 0010)

Страница - 194, смещение EEE = 3822

значение 194 \* 1024 + 3822 = 262144

#include <iostream>

#include <Windows.h>

using namespace std;

#define PAGES 256

#define MB (1024 \* 1024)

#define KB 1024

#define PAGE\_SIZE (KB \* 4) // 4096 KB

int main()

{

const size\_t totalSize = PAGE\_SIZE \* PAGES;

const size\_t arraySize = totalSize / sizeof(int);

LPVOID xmemaddr = VirtualAlloc(NULL, totalSize, MEM\_COMMIT, PAGE\_READWRITE);

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

for (int i = 0; i < arraySize; i++)

array[i] = i;

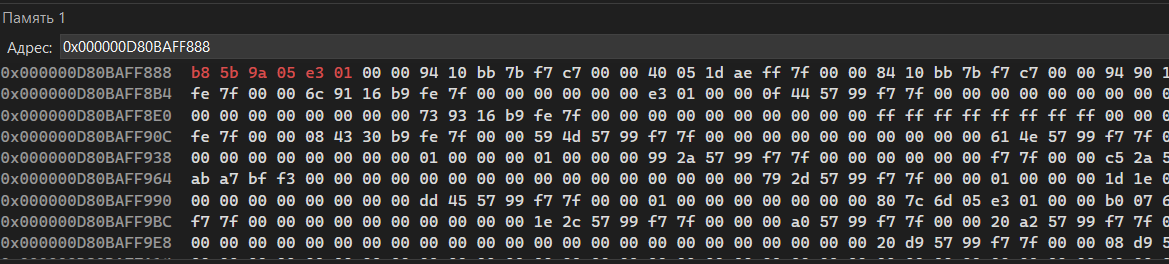
int\* address = array + 194 \* KB + 3822;

if (!VirtualFree(xmemaddr, NULL, MEM\_RELEASE))

cout << "free virtual memory error" << endl;

return 0;

}



**Задание 04**

14-18)

#include <iostream>

#include <Windows.h>

using namespace std;

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" : "")

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

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

<< "\n";

}

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

}

int main()

{

HANDLE pheap = GetProcessHeap();

sh(pheap);

int size = 300000;

int\* m = new int[size];

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

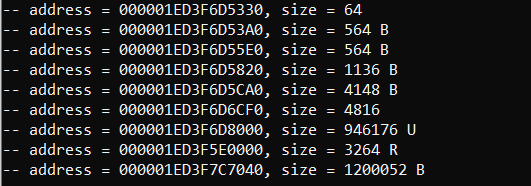
sh(pheap);

system("pause");

return 0;

}





**Задание 05**

19-20)

#include <iostream>

#include <Windows.h>

using namespace std;

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" : "")

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

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

<< "\n";

}

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

}

int main()

{

HANDLE heap = HeapCreate(HEAP\_NO\_SERIALIZE | HEAP\_ZERO\_MEMORY, 1024 \* 1024 \* 4, 0);

sh(heap);

int\* m = (int\*)HeapAlloc(heap, HEAP\_NO\_SERIALIZE | HEAP\_ZERO\_MEMORY, 300000 \* sizeof(int));

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

sh(heap);

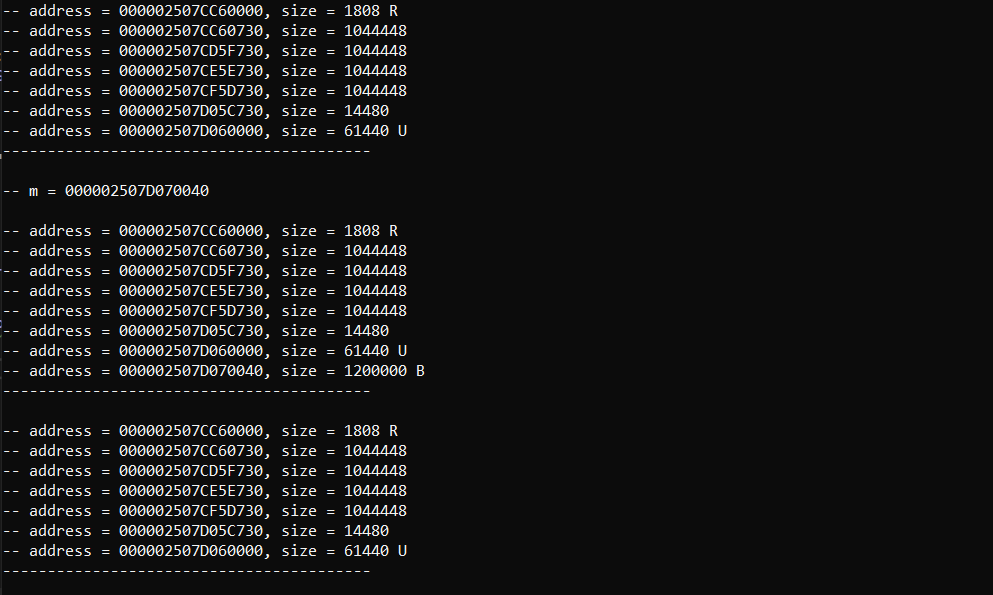
HeapFree(heap, HEAP\_NO\_SERIALIZE, m);

sh(heap);

HeapDestroy(heap);

return 0;

}



**Задание 06**

21-24)

#include <stdio.h>

#include <unistd.h>

int main()

{

pid\_t pid = getpid();

for (int i = 0; i < 10000000; i++) {

printf("Process ID: %d\n", pid);

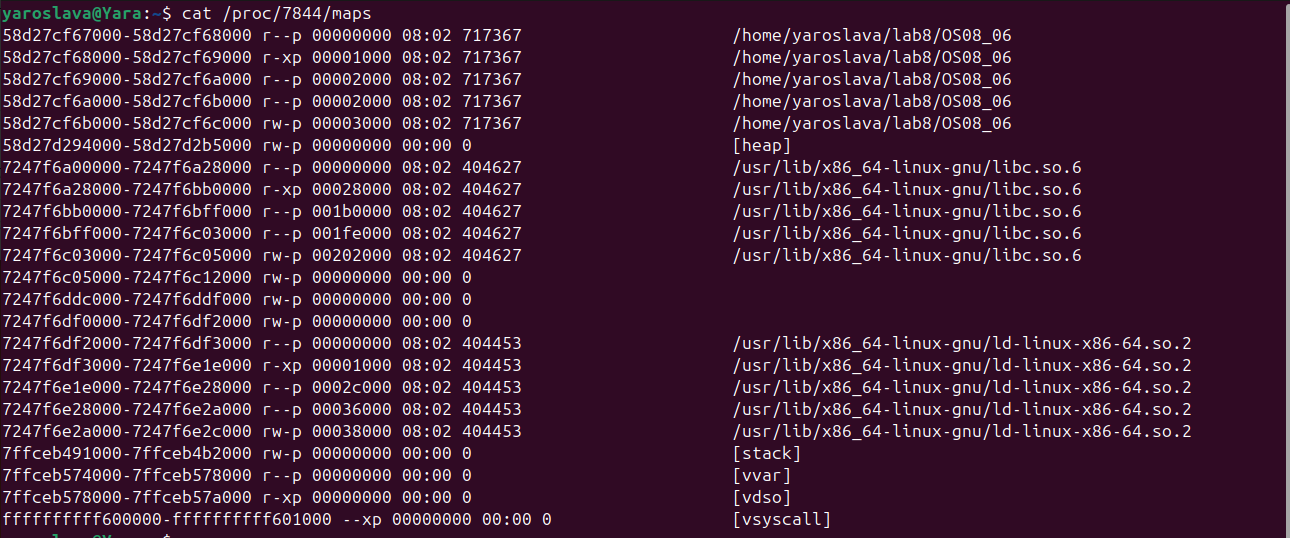
sleep(1);

}

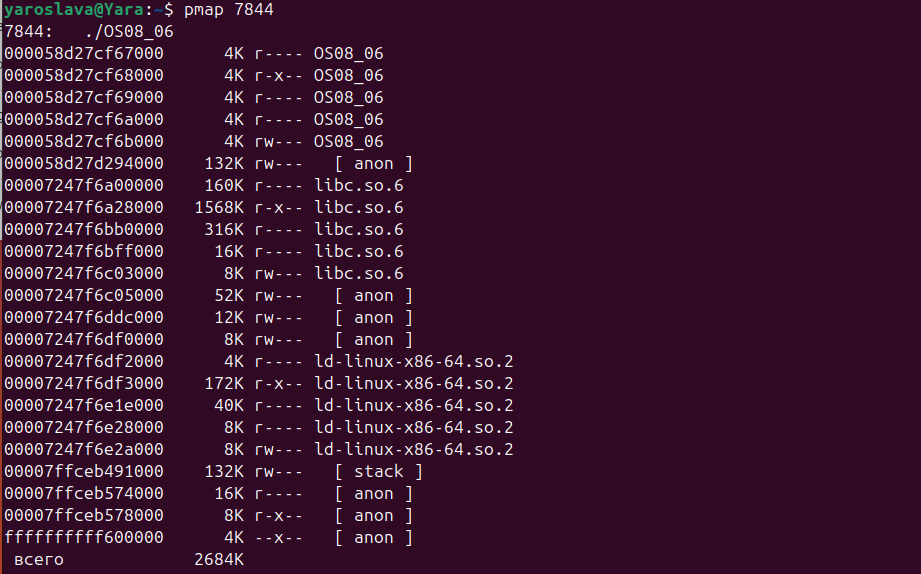
return 0;

}

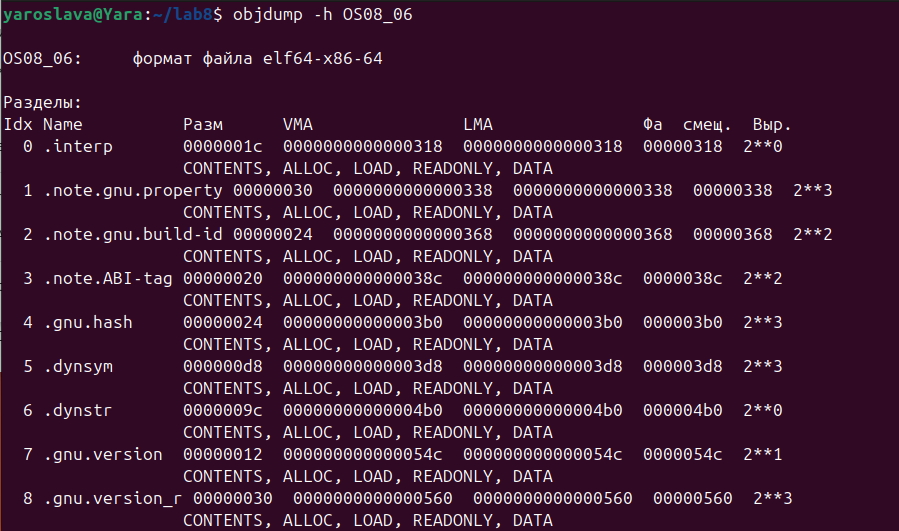
proc

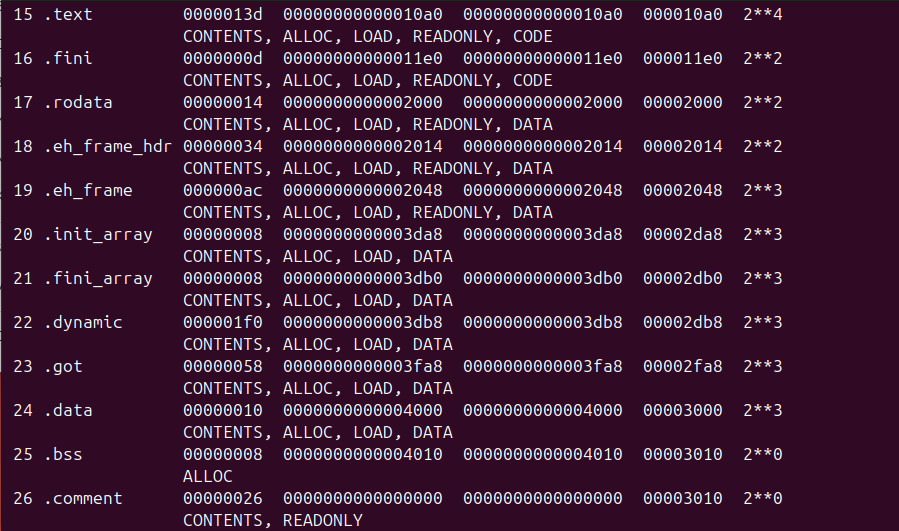


pmap



objdump –h OS08\_06





**Задание 07**

25-29)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

size\_t size = 256 \* 1024 \* 1024;

int\* array = (int\*)malloc(size);

pid\_t pid = getpid();

size\_t arraySize = size / sizeof(int);

for (size\_t i = 0; i < arraySize; i++) {

array[i] = (int)i;

}

printf("Address: %p\n", (void\*)array);

printf("Array size: %zu\n", arraySize);

printf("Process ID: %d\n", pid);

sleep(30);

free(array);

return 0;

}

