**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Дисциплина:**

«Операционные системы»

**Лабораторная работа No2**

“Membomb”

Выполнил:

Хоанг Вьет Зунг – Студент группы N32491

Нгуенг Хонг Ньят - Студент группы N32451

Преподаватель:

Савков С.В.



Санкт-Петербург 2023

**Задание:**

1. Написать программу выделения памяти и заполнения ее нулями с

шагом, равным размеру страницы памяти (mmap, VirtualAlloc)

2. Составить график свободной памяти

3. Ознакомиться с работой демона OOM Killer в Linux

4. Достичь сообщения о невозможности выделить память в Windows

1. **Windows:**

Edition: Windows 7

Ram: 4 GB

**Программа:**

#include <stdio.h>

#include <windows.h>

#include <iostream>

using namespace std;

int main() {

   SYSTEM\_INFO info;

   DWORD PageSize;

   GetSystemInfo(&info);

   PageSize = info.dwPageSize;

   LPVOID p;

   while(true){

    p = VirtualAlloc(NULL, PageSize \* 1024, MEM\_COMMIT | MEM\_RESERVE, PAGE\_EXECUTE\_READWRITE);

    if(p == NULL)cout << "VirtualAlloc failed.\n";

    else{

        memset(p, 0, PageSize \* 1024);

        cout << "Allocating another page.\n";

    }

   }

   return 0;

}

**Code explaination:**

**\*Idea**: This program runs an infinite loop and continuously allocates memory using the VirtualAlloc function from the Windows API. The program first obtains information about the system using the GetSystemInfo function and retrieves the page size in bytes. Then it enters an infinite loop and calls VirtualAlloc twice. The first call to VirtualAlloc with the MEM\_RESERVE flag reserves a contiguous block of memory the size of the page returned by GetSystemInfo, but does not allocate any physical memory. The second call to VirtualAlloc with the MEM\_COMMIT flag commits the previously reserved memory, which means it becomes usable and the actual storage allocated to it. Then the program fills the allocated memory with '0' using the memset function. This program keeps allocating memory in a loop and fills it with '0', which can lead to an out of memory error. Therefore, it is not advisable to allocate memory in an infinite loop without freeing it.

* SYSTEM\_INFO info;: Creates an info variable of type SYSTEM\_INFO, used to store information about the system.
* DWORD PageSize;: Creates a DWORD variable PageSize to store the size of the page
* GetSystemInfo(&info);: Gets information about the system and stores it in the info variable using the GetSystemInfo function.
* PageSize = info.dwPageSize;: Gets the size of the system page and stores it in the variable PageSize.
* LPVOID p;: Creates an uninitialized void pointer.
* while(true){: Starts an infinite loop.
* p = VirtualAlloc(NULL, PageSize \_ 1024, MEM\_COMMIT | MEM\_RESERVE, PAGE\_EXECUTE\_READWRITE);: We use the VirtualAlloc function to allocate a memory block of size PageSize \_ 1024 (1024 pages), returns a void pointer with the capture address header is NULL, flag MEM\_COMMIT | MEM\_RESERVE to ensure that memory is reserved and allocated will be used, PAGE\_EXECUTE\_READWRITE to ensure that memory can be read and written.
* if(p == NULL)cout << "VirtualAlloc failed.\n";: If the returned pointer is NULL, print an error message.
* memset(p, 0, PageSize \_ 1024);: Use the memset function to fill the allocated memory with the character '0' with the size as PageSize \_ 1024.

**Диспетчер задач:**

**\*До запуска:**

**A picture containing text, screenshot, computer, display

Description automatically generated**

**Graphical user interface, application, PowerPoint

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence**

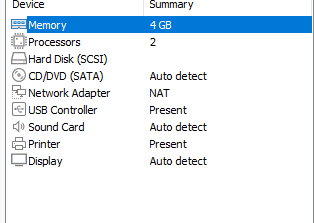
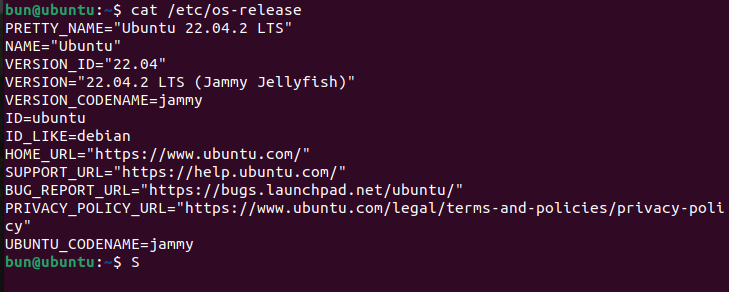
**График свободной памяти в ОС:**

**Chart, line chart, scatter chart

Description automatically generated**

1. **Linux:**

**Name: Ubuntu**

****

**Программа:**

#include <sys/mman.h>

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

int main (){

        unsigned int length = 100\*1024\*1024;

        while(1){

                char \* arr = mmap (0, length, PROT\_READ | PROT\_WRITE, MAP\_PRIVATE | MAP\_ANONYMOUS, 0, 0);

                for(int i=0; i < length; i+= 4096)

                        arr[i]=0;

                system("free -m >> output1.txt");

        }

        return 0;

}

**Code explanation:**

* The program uses the mmap function to allocate a memory area of length 100\*1024\*1024 bytes (equivalent to 100 MB), with read and write permissions (PROT\_READ | PROT\_WRITE).
* PROT\_READ and PROT\_WRITE are flags used to specify access to the memory allocated by the mmap function in the program

+ PROT\_READ allows to read data from the allocated memory

+ PROT\_WRITE allow to write data to the allocated memory.

* To create a memory area that is not associated with any files and is not shared with other processes, the program uses the MAP\_PRIVATE and MAP\_ANONYMOUS options

+ MAP\_PRIVATE specifies that the allocated memory is not shared with other processes, so that any changes to that memory area do not affect other processes using the same memory.

+ MAP\_ANONYMOUS specifies that the allocated memory area is not associated with any files, so that no files are stored on the hard disk corresponding to that memory area.

→ These options in the mmap function specify that the we’re requesting a program allocating an area of memory that is not associated with any files and is not shared with other processes, and allows data to be read and written to that area.

* While (1): The program uses an infinite loop to perform operations on the allocated memory. The program traverses each page of the memory area (each page is 4096 bytes in size) and assigns the value 0 to the first element of each page. Assigning this value is to ensure that the page is loaded into physical memory (rather than just stored on the hard disk) and to ensure that OOM Killer will release these pages first when the system is Out of Memory.
* Finally, we use the system command to execute the "free -m" command and writes the results to the output1.txt file. The "free -m" command is used to display information about the memory and swap space used on the system. For each loop, the results are written to the output1.txt file to monitor memory usage on the system.

**Диспетчер задач:**

**Graphical user interface

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**График свободной памяти в ОС: Chart, line chart

Description automatically generated**

1. **Демон OOM Killer в Linux**

* OOM (Out of Memory) Killer is a feature of the Linux operating system, designed to solve the problem when the system encounters Out Of Memory condition - ie the system has used up all RAM and swap space.
* When an OOM condition occurs, the Linux operating system will try to free memory by running OOM Killer. OOM Killer is a system process, designed to find and "kill" the processes that are using the most memory, in order to free up memory for the system.
* OOM Killer's "kill" selection process is determined by the OOM score system. This score is calculated as the total amount of memory used by the process, divided by the amount of time that process has been running. That is, if a process uses a lot of memory but runs for a short time, its score will be higher than a process that uses little memory but runs continuously for a long time.
* Check the priority of the processes: OOM Killer will check the priority of the processes running on the system. Processes with lower priority will be given priority to kill before other processes.
* The out\_of\_memory() function in OOM Killer has the main role of handling the case that the system encounters an Out-of-Memory (OOM) condition. When the system is running low on memory, the kernel will call this function to identify and free memory from unnecessary or lowest-priority processes.
* Specifically, the out\_of\_memory() function will call select\_bad\_process(), select\_victim\_process() and oom\_kill\_process(), respectively, to identify and kill the process to free up memory. If no processes are found to kill, this function will log and send an alert to applications to report the OOM status on the system. then the badness() function is used to calculate the priority of a process when the system encounters an Out-of-Memory (OOM) condition. This function will calculate a "badness score" value based on factors like(Memory Allotted Size of Process, Process Priority, Process CPU Time, Count amount of swap memory used by the process,). These factors will affect the priority of the process when the system decides to kill a process to free up memory. Processes with higher "badness score" are likely to be killed first
* Identify the parent process: If a parent process and one or more of its child processes are running on the system, OOM Killer will give priority to killing the child process instead of the parent process.
* However, sometimes "killing" processes can have undesirable consequences, such as disrupting the operation of important applications. So configuring OOM Killer to ensure system stability and performance is very important.

1. **Get the message about not being able to allocate memory in Windows**

I was running the program on Windows 7 and but I could not get the message about not being able to allocate memory, the only message was "VirtualAlloc." which are written in the program in case the VirtualAlloc() function doesn't work. The system sometimes gets frozen and after while it turned off by itself.