# ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

# Факультет безопасности информационных технологий Дисциплина:

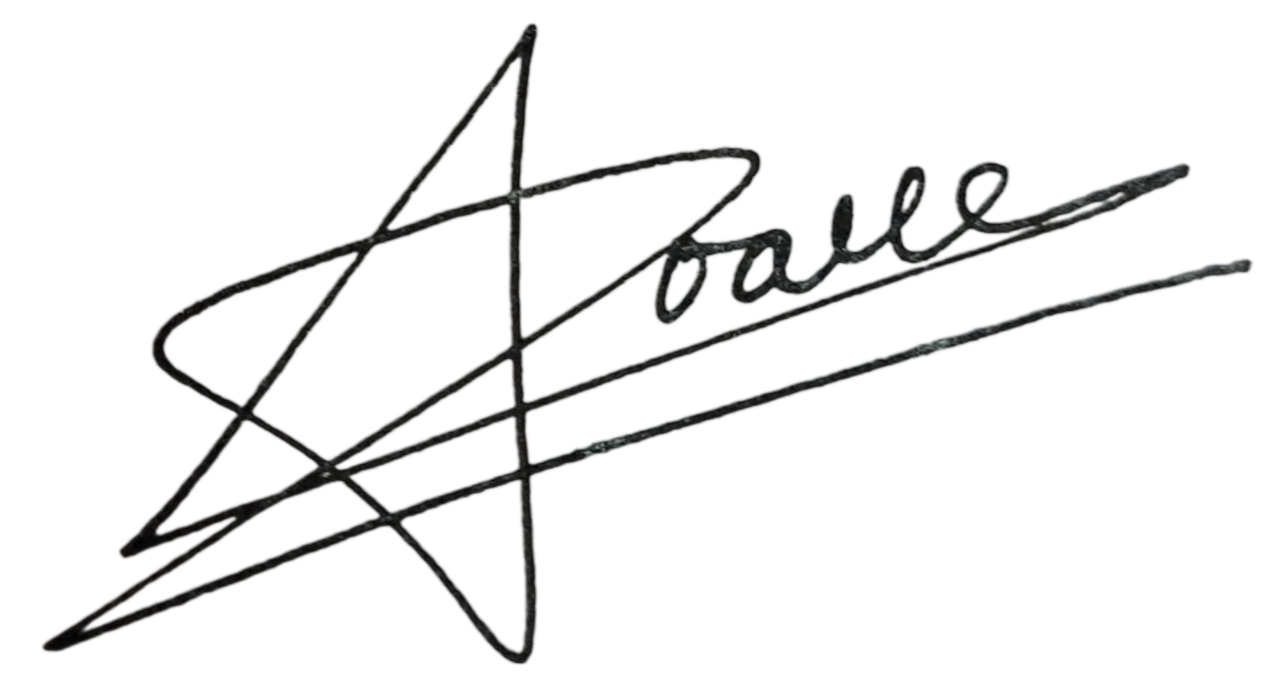
«Операционные системы»

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

«Membomb»

# Выполнил:

Чу Ван Доан N3247





(подпись)

# Проверил:

Савков Сергей Витальевич



(подпись)

Санкт-Петербург

# 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

# Написать программу выделения памяти и заполнения ее нулями с шагом, равным размеру страницы памяти (mmap, VirtualAlloc).

* 1. **Linux (Ubuntu 22.04 LTS)**

**1. Написать программу выделения памяти и заполнения ее нулями с шагом, равным размеру страницы памяти (mmap, VirtualAlloc)**

**Программа:**

**Это программа, работающая в Linux, написанная на языке программирования C.**

#include <stdlib.h>

#include <sys/mman.h>

#include <string.h>

#include <unistd.h>

int main(){

unsigned int size = 50\*1024\*1024; Lệnh này khai báo một biến có tên size và khởi tạo nó thành 50 megabyte. Эта команда объявляет переменную с именем size и инициализирует ее размером 50 мегабайт.

while(1){

unsigned char \*p = mmap(NULL, size, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, 0,0);

Lệnh này gọi hàm mmap() để ánh xạ một vùng bộ nhớ vào không gian địa chỉ của chương trình. Hàm mmap() có các đối số sau: Эта команда вызывает функцию mmap() для отображения области памяти в адресное пространство программы. Функция mmap() имеет следующие аргументы:

Địa chỉ tại đó ánh xạ vùng bộ nhớ. Nếu đối số này là NULL, hệ điều hành sẽ chọn một địa chỉ. Адрес, по которому отображается область памяти. Если этот аргумент равен NULL, операционная система выбирает адрес.

* len: Kích thước của vùng bộ nhớ cần ánh xạ. len: Размер области памяти для сопоставления.
* prot: Cờ bảo vệ cho vùng bộ nhớ. Trong trường hợp này, vùng được ánh xạ với cả quyền đọc và ghi. prot: Флаг защиты области памяти. В этом случае зона отображается с разрешениями как на чтение, так и на запись.
* flags: Các cờ chỉ định các vùng được ánh xạ. Trong trường hợp này, vùng được ánh xạ riêng tư và ẩn danh. флаги: Флаги определяют отображаемые регионы. В этом случае отображаемая зона является частной и анонимной.
* fd: Số mô tả tệp của tệp chứa vùng bộ nhớ cần ánh xạ. Trong trường hợp này, vùng bộ nhớ không được ánh xạ từ tệp, vì vậy đối số này được đặt thành 0. fd: номер дескриптора файла, содержащего область памяти, подлежащую сопоставлению. В этом случае область памяти не отображается из файла, поэтому этому аргументу присваивается значение 0.
* offset: Vị trí trong tệp tại đó ánh xạ vùng bộ nhớ. Trong trường hợp này, vùng bộ nhớ không được ánh xạ từ tệp, vì vậy đối số này được đặt thành 0.
* offset: место в файле, куда отображается область памяти. В этом случае область памяти не отображается из файла, поэтому этому аргументу присваивается значение 0.

Hàm mmap() trả về một con trỏ đến vùng bộ nhớ được ánh xạ. Функция mmap() возвращает указатель на отображаемую область памяти.

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

p[i] = 0;

Lệnh này đánh dấu đầu vòng lặp for lặp qua vùng bộ nhớ được ánh xạ, viết một số 0 cho mỗi byte. Эта команда отмечает начало цикла for, который проходит по отображенной области памяти, записывая ноль в каждый байт.

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

}

return 0;

}

Nhìn chung, chương trình này tạo một vòng lặp vô hạn lặp đi lặp lại ánh xạ một vùng bộ nhớ 50 megabyte vào không gian địa chỉ của chương trình và viết các số 0 vào tất cả các byte trong vùng. Chương trình sau đó gọi hàm system() để thực thi lệnh free, hiển thị lượng bộ nhớ trống và sử dụng trên hệ thống. Đầu ra của lệnh free được thêm vào tệp membomb.txt.

Chương trình này là một loại tấn công từ chối dịch vụ, vì nó khiến hệ thống hết bộ nhớ trống. Nếu chương trình được phép chạy trong một thời gian dài, cuối cùng nó có thể khiến hệ thống bị treo

В целом, **эта программа создает итеративный бесконечный цикл, который отображает 50-мегабайтную область памяти в адресное пространство программы и записывает нули во все байты в этой области. Затем программа вызывает функцию system() для выполнения команды free, отображая объем свободной и использованной памяти в системе. Вывод команды free добавляется в файл membomb.txt.**

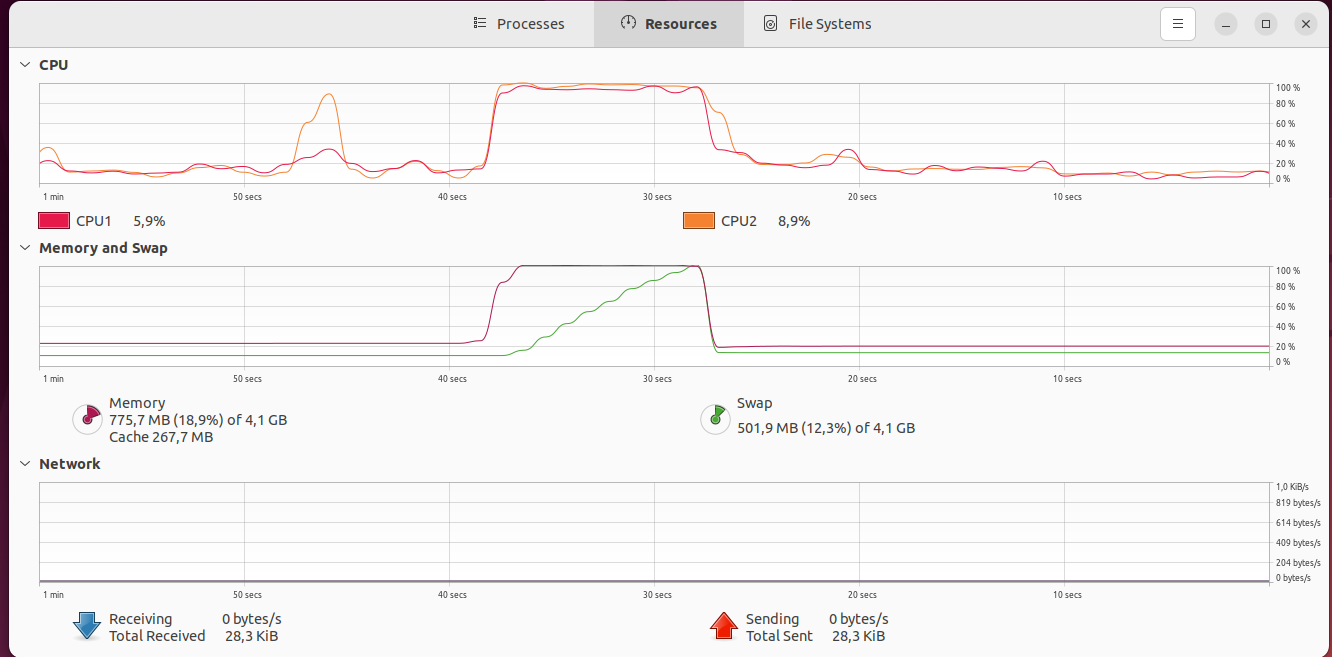
**Эта программа представляет собой разновидность атаки типа «отказ в обслуживании», поскольку она приводит к нехватке свободной памяти в системе. Если программе разрешено работать в течение длительного времени, это может в конечном итоге привести к сбою системы.**

**2. Составить график свободной памяти**

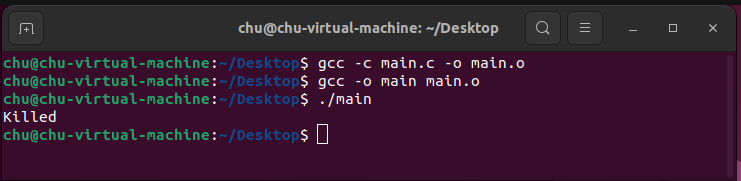
**- До запуска программы:**



**- После запуска программы:**

****

**- ООМ**

****

**Это программа, работающая в Linux, написанная на языке программирования C.**

# Windows

**1. Написать программу выделения памяти и заполнения ее нулями с шагом, равным размеру страницы памяти (mmap, VirtualAlloc)**

**Программа:**

#include <stdlib.h> #include <windows.h> #include <cstring> #include <unistd.h> using namespace std; int main() {

SYSTEM\_INFO si;

GetSystemInfo(&si);

int size = si.dwPageSize;

void\* ptr; while (1){

LPVOID ptr = VirtualAlloc(NULL, size, MEM\_RESERVE, PAGE\_READWRITE); ptr = VirtualAlloc(ptr, size, MEM\_COMMIT, PAGE\_READWRITE); memset(ptr, '0', size);

}

return 0;

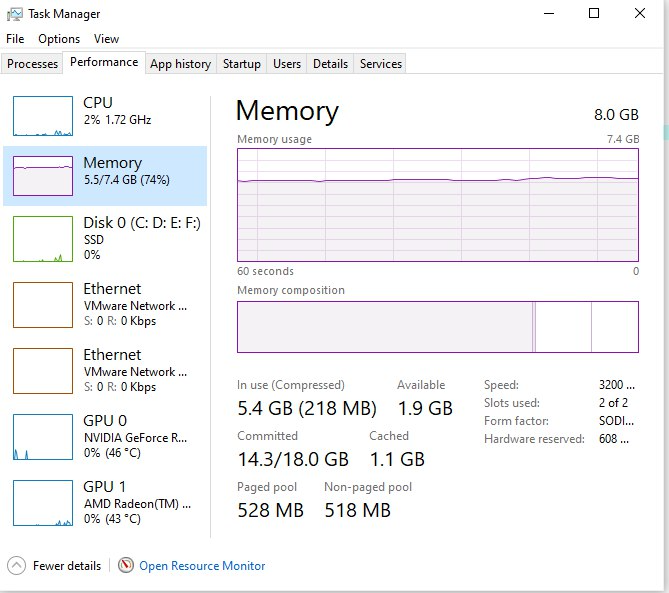
}

đoạn mã trên sẽ tạo ra một vòng lặp vô hạn, trong đó mỗi lần lặp vòng chương trình sẽ cấp phát một vùng bộ nhớ kích thước bằng kích thước trang bộ nhớ của hệ thống. Sau đó, chương trình sẽ gán giá trị '0' cho tất cả các byte trong vùng bộ nhớ được cấp phát. Cuối cùng, chương trình sẽ giải phóng vùng bộ nhớ đã được cấp phát.

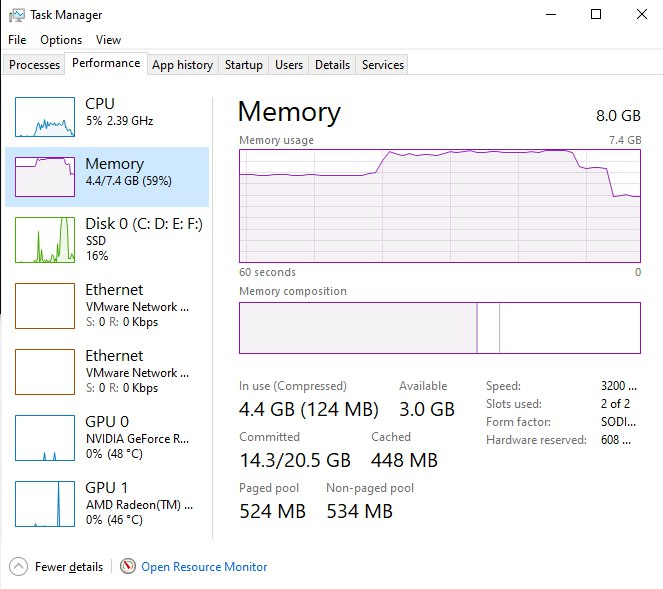
**Приведенный выше код создает бесконечный цикл, в котором на каждой итерации программы будет выделяться область памяти, равная размеру страницы памяти системы. Затем программа присвоит значение «0» всем байтам в выделенной области памяти. Наконец, программа освободит выделенную область памяти.**

**2. Составить график свободной памяти**

**- До запуска программы:**



**- После запуска программы:**



**2. Ознакомиться с работой демона OOM Killer в Linux**

Когда у сервера или процесса заканчивается память, Linux предлагает 2 пути решения: обрушить всю систему или завершить процесс (приложение), который съедает память. Лучше, конечно, завершить процесс и спасти ОС от аварийного завершения. В двух словах, Out-Of-Memory Killer (OOM Killer)— это процесс, который завершает приложение, чтобы спасти ядро от сбоя. Он жертвует приложением, чтобы сохранить работу ОС.

Khi một máy chủ hoặc tiến trình hết bộ nhớ, Linux đưa ra 2 giải pháp: đánh sập toàn bộ hệ thống hoặc chấm dứt tiến trình (ứng dụng) đang ngốn bộ nhớ. Tất nhiên, tốt hơn hết là bạn nên chấm dứt quá trình này và cứu hệ điều hành khỏi gặp sự cố. Tóm lại, Out-Of-Memory Killer (OOM Killer) là một quá trình giết chết một ứng dụng để cứu kernel khỏi bị treo. Nó hy sinh ứng dụng để giữ cho hệ điều hành hoạt động

OOM Killer - это компонент ядра Linux, призванный решать проблему недостатка памяти. Известно, что виртуальной памяти может быть бесконечно много (в пределах адресации), а вот физической - вполне конечное число. Ядро выделяет память процессам "с запасом" в сумме превышающую физическую память системы. В основном, всё разруливается нормально (вся выделенная память одновременно редко требуется), но бывает ситуация когда становится нужно памяти больше, чем ее физически есть. И системе тогда нужно завершить какой-то процесс, чтобы продолжить работу. Вот этим и занимается OOM Killer.

OOM Killer là một thành phần của nhân Linux được thiết kế để giải quyết vấn đề bộ nhớ. Được biết, có thể có số lượng bộ nhớ ảo vô hạn (trong giới hạn địa chỉ), nhưng bộ nhớ vật lý có thể là một con số khá hữu hạn. Hạt nhân phân bổ bộ nhớ cho các tiến trình "có giới hạn" với số lượng vượt quá bộ nhớ vật lý của hệ thống. Về cơ bản, mọi thứ đều diễn ra bình thường (tất cả bộ nhớ được phân bổ hiếm khi cần thiết cùng một lúc), nhưng có một tình huống khi cần nhiều bộ nhớ hơn mức có sẵn về mặt vật lý. Và hệ thống sau đó cần phải chấm dứt một số quy trình để tiếp tục hoạt động. Đây là những gì OOM Killer làm.

Когда заканчивается память, вызывается функция out\_of\_memory(). В ней есть функция select\_bad\_process(), которая получает оценку от функции badness(). Под раздачу попадет самый «плохой» процесс. Функция badness() выбирает процесс по определенным правилам.

Khi hết bộ nhớ, hàm out\_of\_memory() sẽ được gọi. Nó có hàm select\_bad\_process() nhận đánh giá từ hàm badness(). Quá trình “tồi tệ nhất” sẽ được nhắm mục tiêu. Hàm badness() chọn một tiến trình theo những quy tắc nhất định.

1. Ядру нужен какой-то минимум памяти для себя.

2. Нужно освободить много памяти.

3. Не нужно завершать процессы, которые используют мало памяти.

4. Нужно завершить минимум процессов.

5.Сложные алгоритмы, которые повышают шансы на завершение для тех процессов, которые пользователь сам хочет завершить.

1. Hạt nhân cần một lượng bộ nhớ tối thiểu cho chính nó.

2. Bạn cần giải phóng nhiều bộ nhớ.

3. Không cần phải chấm dứt các tiến trình sử dụng ít bộ nhớ.

4. Cần phải hoàn thành các quy trình tối thiểu.

5. Các thuật toán phức tạp giúp tăng cơ hội hoàn thành cho những quy trình mà bản thân người dùng muốn hoàn thành.

Выполнив все эти проверки, OOM изучает оценку (oom\_score). OOM назначает oom\_score каждому процессу, а потом умножает это значение на объем памяти. У процессов с большими значениями больше шансов стать жертвами OOM Killer. Процессы, связанные с привилегированным пользователем, имеют более низкую оценку и меньше шансов на принудительное завершение.

Всякий раз, когда OOM Killer вызывается для уничтожения процесса, он записывает информацию в системный журнал, включая информацию о том, какой процесс был убит и почему. Проверяем следующее: **dmesg | egrep -i “killed process”**

**IV. Достичь сообщения о невозможности выделить память в Windows**

- **После первого запуска программы membomb в Windows, как и на графике, память**

**выделяется и используется очень быстро, и программа израсходовала память, компьютер**

**начинает зависать, и операционная система немедленно реагирует, закрывая программу**

**и возвращая память.**

**И после этого компьютер продолжает нормально работать.**

**- После перезагрузки компьютера и повторного запуска программы membomb память**

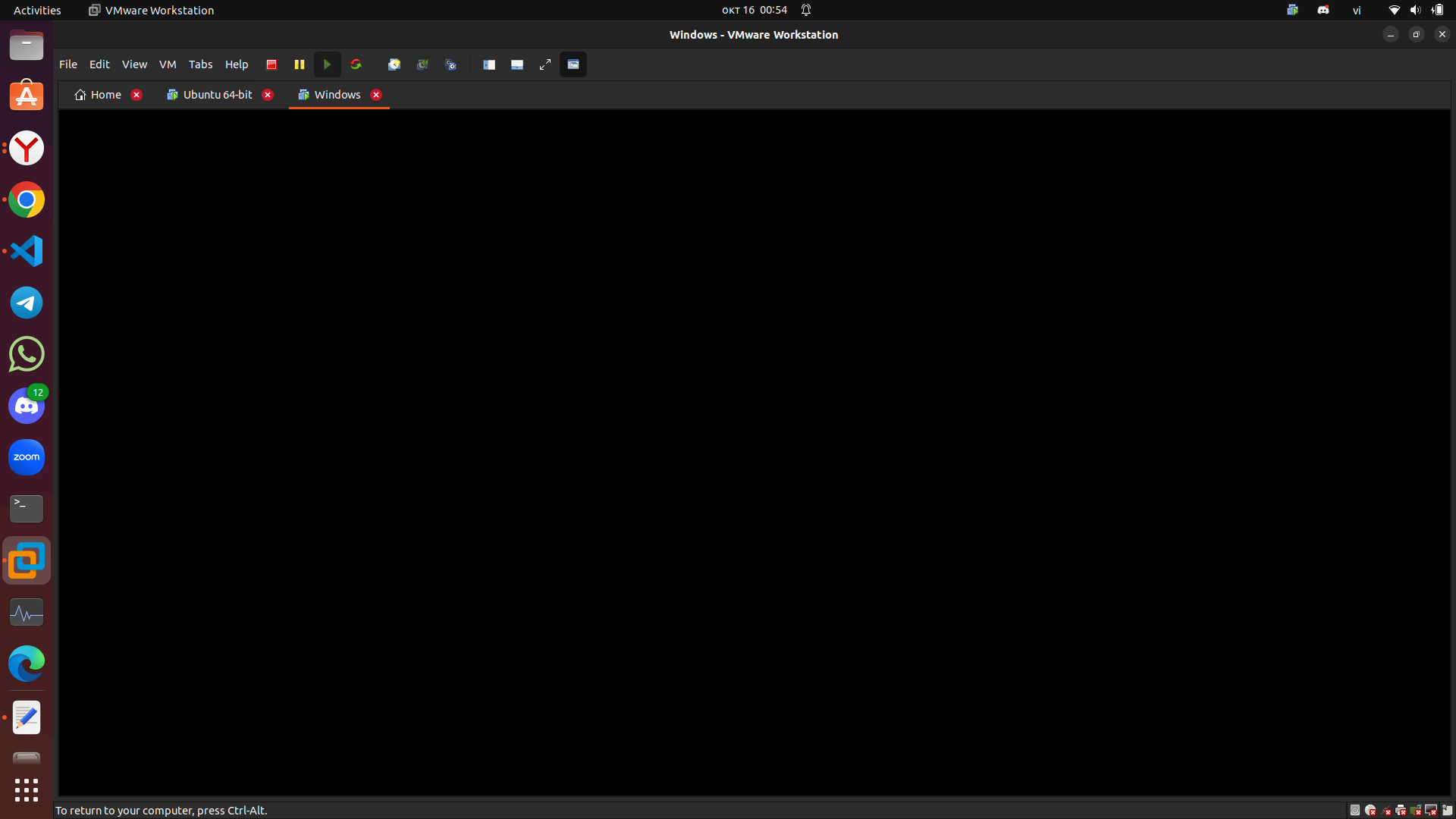
**выделялась и использовалась очень быстро, и программа израсходовала всю память,**

**компьютер начал зависать и сразу экран становился черным . Я ничего не мог сделать**

**с компьютером, и мне пришлось перезагрузить компьютер.**

Sau lần đầu tiên bạn chạy chương trình membomb trên Windows, như trong biểu đồ, bộ nhớ được cấp phát và sử dụng rất nhanh và chương trình hết bộ nhớ, máy tính bắt đầu treo và hệ điều hành ngay lập tức phản ứng bằng cách đóng chương trình và trả lại kí ức. Và sau đó máy tính vẫn tiếp tục hoạt động bình thường.

- Sau khi khởi động lại máy tính và chạy lại chương trình membomb, bộ nhớ được cấp phát và sử dụng rất nhanh, đồng thời chương trình đã dùng hết bộ nhớ, máy tính bắt đầu đơ và ngay lập tức màn hình chuyển sang màu đen. Tôi không thể làm gì với máy tính và phải khởi động lại máy tính.



**Вывод:** При выделении памяти в windows с помощью функции VirtualAlloc(). Свободная память резко упадет. При выделении памяти в Linux (функция mmap())свободная память будет медленно уменьшаться, потому что в ядре Linux есть OOM Killer, поэтому, когда памяти выделяется слишком много, вызывая переполнение физической памяти системы, OOM отключит этот процесс.

Khi cấp phát bộ nhớ trong windows bằng hàm VirtualAlloc(). Bộ nhớ trống sẽ giảm mạnh. Khi cấp phát bộ nhớ trong Linux (hàm mmap()), bộ nhớ trống sẽ giảm dần do nhân Linux có OOM Killer nên khi cấp phát quá nhiều bộ nhớ khiến bộ nhớ vật lý của hệ thống đầy, OOM sẽ kill tiến trình.