**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Операционные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

«Membomb»

Содержание

[Введение 4](#_Toc178428403)

[1 MEMBOMB для Linux, Windows 5](#_Toc178428404)

[1.1 Задание 5](#_Toc178428405)

[1.2 Ход работы 5](#_Toc178428406)

[1.2.1 Membomb для Linux 5](#_Toc178428407)

[1.2.2 Membomb для Windows 6](#_Toc178428408)

[1.2.3 График свободной памяти 6](#_Toc178428409)

[1.2.4 Демона OOM Killer в Linux 8](#_Toc178428410)

[Заключение 10](#_Toc178428411)

[Список использованных источников 11](#_Toc178428412)

Введение

Цель работы – познакомиться и реализовать программу Membomb

Для достижения поставленной цели необходимо решить следующие задачи:

* написать программу выделения памяти и заполнения ее нулями с

шагом, равным размеру страницы памяти (mmap, VirtualAlloc);

* составить график свободной памяти;
* ознакомиться с работой демона OOM Killer в Linux;
* достичь сообщения о невозможности выделить память в Windows.

# MEMBOMB для Linux, Windows

Membomb – это программа, которая заполняет выделенную память различными данными и не освобождает место, до тех пор пока места не закончится.

## Задание

Написать membomb для Linux, Windows. Составить график свободной памяти. Ознакомиться с работой демона OOM Killer в Linux.

## Ход работы

### Membomb для Linux

Код membomb для линкс представлен в листинге 1. Для работы с памятью мы использовали mmap. Сначала с помощью функции sysconf(\_SC\_PAGESIZE) получаем размер страницы памяти. Затем открываем бесконечный цикл, в котором выделяем память с помощью функции mmap(). Далее мы заполняем выделенную область памяти нулями. Для этого мы используем цикл, который проходит по всем байтам страницы и устанавливает их значение в 0.

 #include <stdio.h> // ввод-вывод

#include <stdlib.h> //функции динамического выделения памяти

#include <unistd.h> //системные функции

#include <sys/mman.h> //функции управления памятью

int main() {

    // Получаем размер страницы памяти

    long page\_size = sysconf(\_SC\_PAGESIZE);

    while (1) {

        // Выделяем память размером в 1 страницу

        void \*ptr = mmap(NULL, page\_size, PROT\_READ | PROT\_WRITE, MAP\_PRIVATE | MAP\_ANONYMOUS, -1, 0);

        if (ptr == MAP\_FAILED) {

            perror("mmap");

            return 1;

        }

        // Заполняем память нулями

        for (int i = 0; i < page\_size; i++) {

            ((char \*)ptr)[i] = 0;

        }

        // Не освобождаем память

    }

    return 0;

}

1. Membomb для Linux.

После запуска мембомбы используемая память резко начинает расти, система начинает работать медленнее, немного зависать. Но через некоторое время срабатывает OOM Killer, оперативная память освобождается. На приведенном ниже скриншоте можно увидеть, что мембомба была убита OOM Killer.

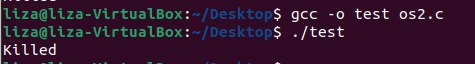


Рисунок 1 - терминал после запуска мембомбы.

### Membomb для Windows

Код membomb для виндоус представлен в листинге 2.

#include <windows.h>

#include <stdio.h>

#include <errno.h>

int main () {

while (1) {

LPVOID memory = calloc(4096, 8);

if ( memory != NULL ) {

memset ( memory , 0 , 1) ;

}

else {

fprintf ( stderr , "% s " , strerror ( errno ) ) ;

}

}

getchar () ;

return 0;

}

Листинг 2 – membomb для windows.

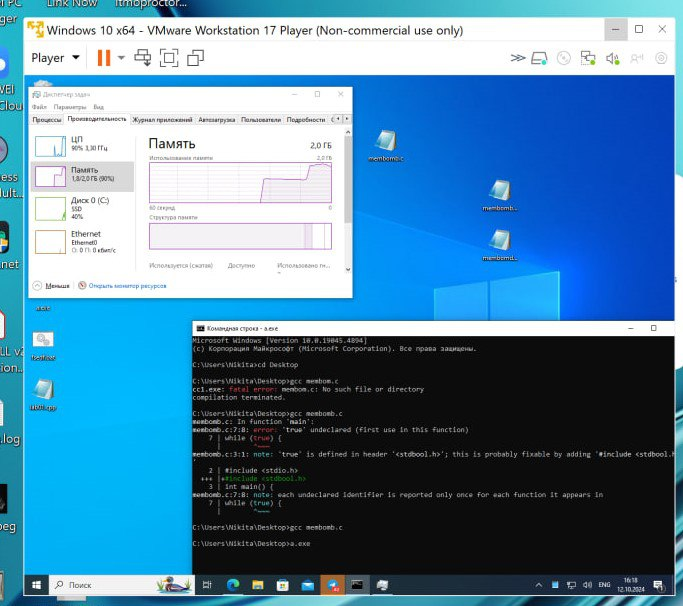


Рисунок 2 - терминал и используемая память после запуска мембомбы.

После запуска мембомбы используемая память резко начинает расти, система начинает работать медленнее, зависать. Потом появляется черный экран.

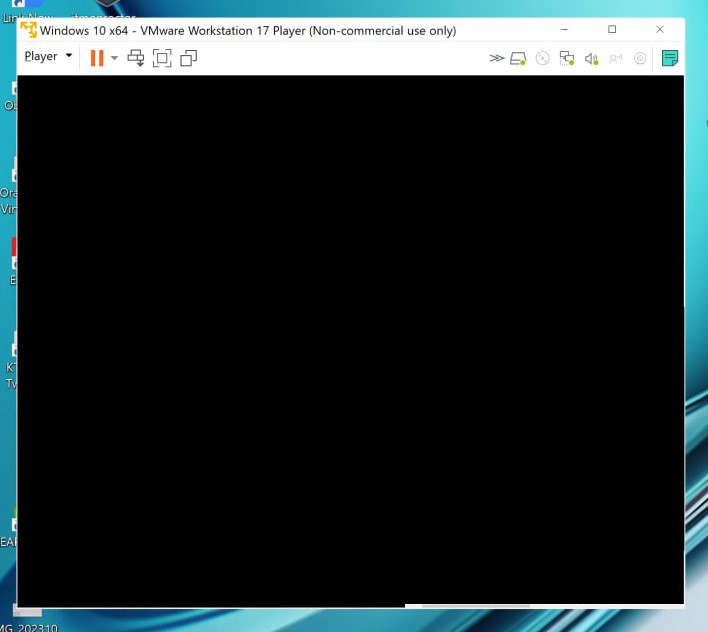


Рисунок 3 – черный экран.

### График свободной памяти

Для того, чтобы составить график свободной памяти для ос линукс, перед запуском мембомбы в терминале была использована команда команду sudo sar -r 1 30 для сбора данных о свободной памяти каждую секунду в течение 30 секунд. После была запущена мембомба. На основе полученных данных был составлен приведенный ниже график.

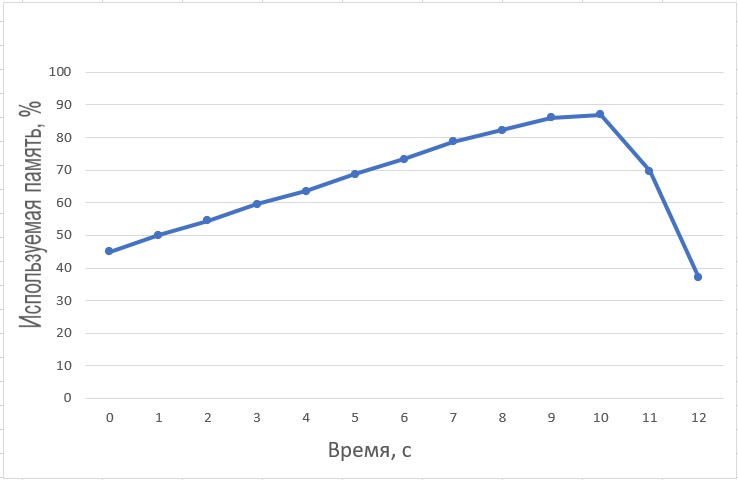


Рисунок 4 – график свободной памяти linux.

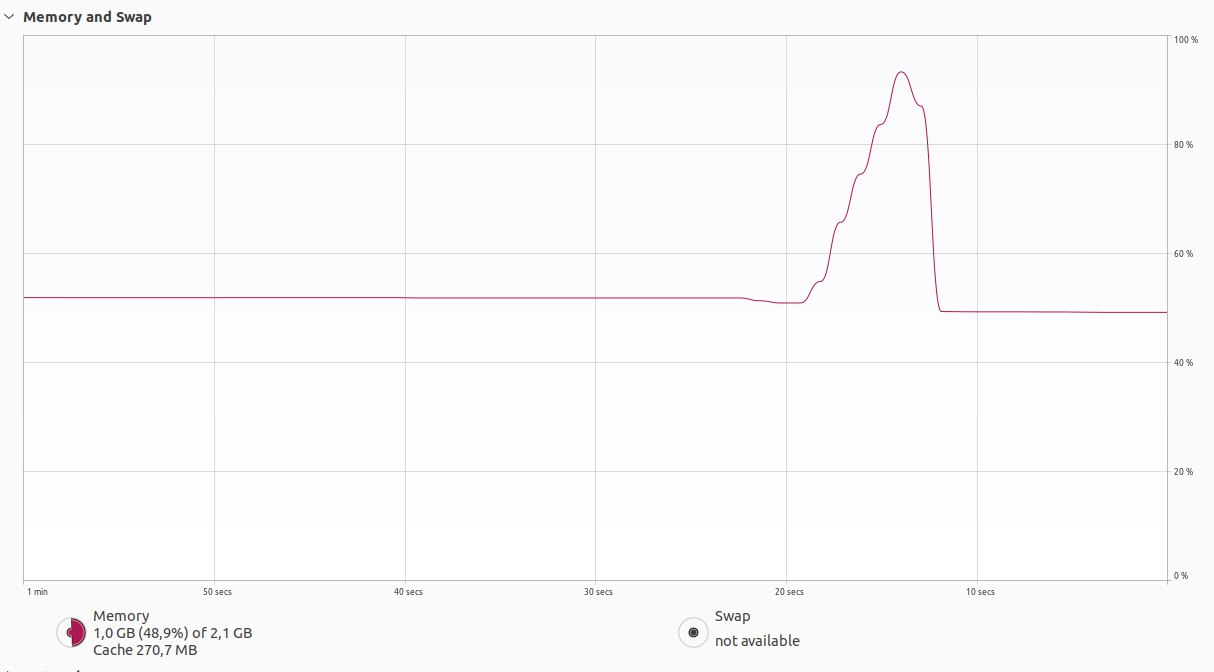


Рисунок 5 – диспетчер задач в линукс после запуска мембомбы.

Для того чтобы составить график свободной памяти для ос виндоус, перед запуском мембомбы был открыт диспетчер задач, а в нем раздел с памятью, в этом раздели можно увидеть свободную память. Одновременно с запуском мембомбы велась видеосъемка экрана. После завершения эксперимента, на записанном нами видео можно было увидеть, как изменяется количество свободной памяти со временем. По полученным данным и был составлен приведенный ниже график.

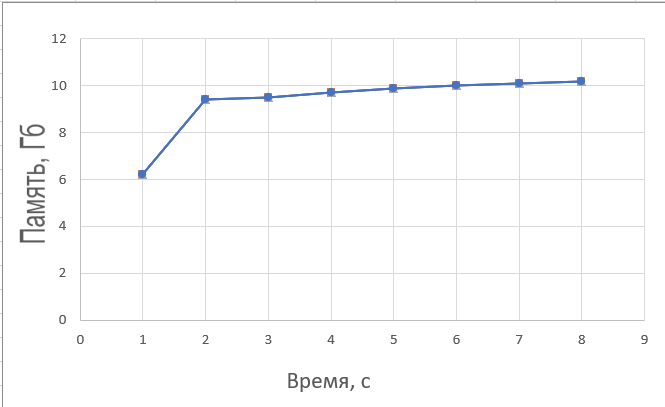


Рисунок 6 – график свободной памяти windows.

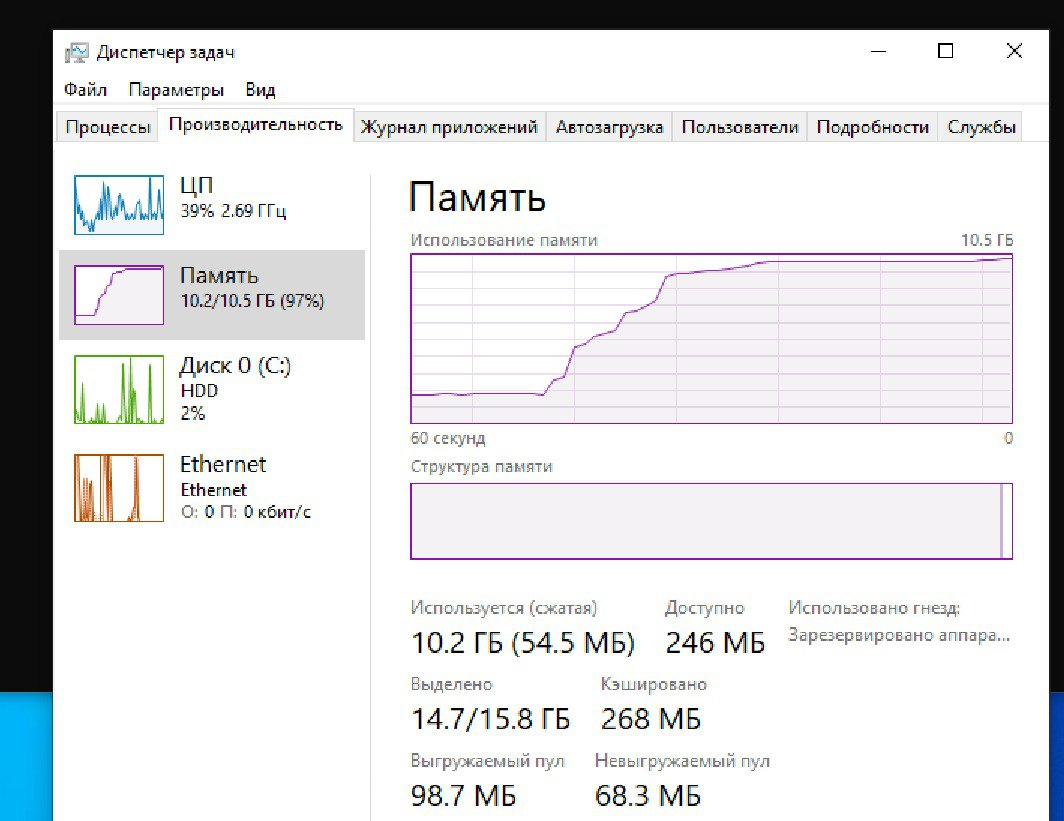


Рисунок 7 – диспетчер задач в виндоус после запуска мембомбы.

### Демона OOM Killer в Linux

OOM Killer — механизм ядра Linux, который при исчерпании доступной памяти принудительно завершает отдельные процессы на сервере для освобождения оперативной памяти (RAM). Освободившаяся память передается ядру ОС, а после перенаправляется тому процессу, которому ее было недостаточно.

OOM killer убивает процессы сигналом SIGKILL, не предоставляя им возможность корректно завершить свое выполнение (сохранить данные, вызвать другие процессы). Поэтому результат работы OOM killer часто приводит к тяжелым последствиям.

Чтобы завершить процесс, Linux вызывает функцию out\_of\_memory. В функции out\_of\_memory вызывается функция select\_bad\_process, которая позволяет выбрать процесс для завершения. Все запущенные процесс оцениваются с помощью функции badness.

После убийства, в системном журнале появится строчка: Out of Memory: Killed process 2256 (mysql) score 907 or sacrifice child.

OOM Killer использует достаточно сложный анализ и присваивает каждому процессу очки негодности (репутация) (badness). При исчерпании памяти, будет убит именно самый негодный процесс. Очки негодности могут принимать значения от -1000 до 1000, чем выше значение, тем более высока вероятность что OOM Killer убьет процесс.

На очки негодности влияют такие показатели как дочерние процессы, имеющие собственную виртуальную память; приоритет процесса; процессорное время (чем более активен процесс и чем больше процессорного времени он использует); время жизни процесса; от какого имени запущенны процессы (например, root) и т.д. Также можно увеличить, или уменьшить репутацию процесса, добавив в файл oom\_adj значение от -16 до +15.

Количество очков можно узнать с помощью команды: cat /proc/PID/oom\_score

Как правило, в первую очередь завершаются недавно запущенные пользовательские процессы, которые требуют большого объема памяти и/или имеют множество дочерних процессов; в последнюю очередь завершаются системные процессы пользователя root, запущенные значительное время назад.

При желании можно отключить OOM Killer, но это крайне не рекоммендуется. Сделать это можно следующим образом: $ sudo cat /proc/sys/vm/panic\_on\_oom

По умолчанию тут указан 0. Это значит, что OOM Killer включен. Чтобы отключить его: $ sudo echo 1 > /proc/sys/vm/panic\_on\_oom.

Заключение

В ходе данной лабораторной работы были реализованы membomb для операционных систем linux и windows. В результате реализации мы изучили поведение систем при заполнении памяти, что позволило лучше понять управление памятью в ОС. Таким образом мы смогли выяснить некоторые различия в работе ОС. Также мы познакомились с работой демона OOM Killer. Это позволило закрепить полученные знания и навыки. В ходе выполнения лабораторной работы были выполнены все задачи и достигнуты поставленные цели.

Список использованных источников

1. Использование Out Of Memory Killer в Linux // VMBlog. URL: https://vmblog.ru/out-of-memory-killer-v-linux/#:~:text=Out%20of%20Memory%20Killer%20(OOM,может%20предоставить%20ее%20другим%20процессам.
2. Linux - начинающим. Что такое OOM Killer и как он работает // interface31. URL: <https://interface31.ru/tech_it/2022/09/linux---nachinayushhim-chto-takoe-oom-killer-i-kak-on-rabotaet.html>.
3. Как работает OOM Killer // дзен. URL: <https://dzen.ru/a/Zb-WHGyGjUbgy7BDushhim-chto-takoe-oom-killer-i-kak-on-rabotaet.html>.
4. Эндрю Таненбаум Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с.