Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра обчислювальної техніки

Лабораторна робота 5

З дисципліни «Архітектура компʼютерів 2»

ВИКОНАВ:

студент ІІІ курсу ФІОТ

Київ-2023

**Завдання:**

I. Додайте BUG\_ON() замість друку повідомлення та повернення -

EINVAL для неприпустимого значення параметра.

II. Додайте примусове внесення помилки “начебто kmalloc() повернув

0” під час формування елемента списку для якогось повідомлення

(останнього із серії, 5-го, ... — на ваш вибір).

III. Модифікуйте Makefile аналогічно appendix1.

IV. Отримайте обидва повідомлення, роздивіться їх та для одного

з них виконайте пошук місця аварії аналогічно appendix1.

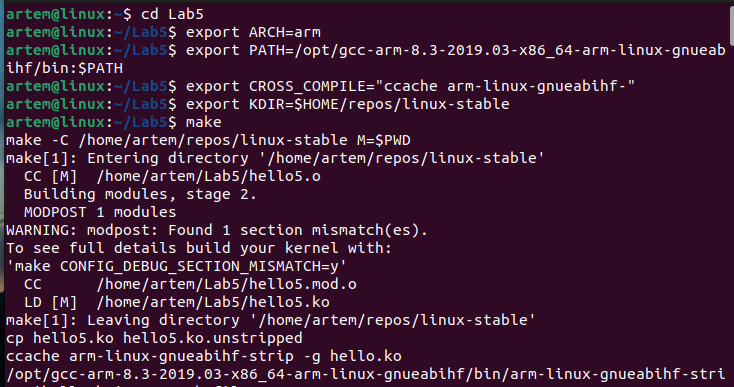
A. Зауважте, що при виконанні BUG\_ON() модуль буде

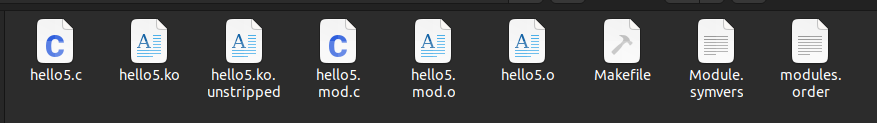
“зайнятий”, і ви не зможете виконати rmmod.

**GitHub:**

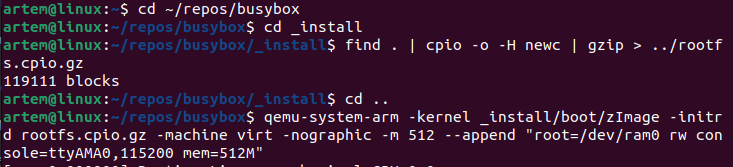
**Хід роботи**

Створення необхідних файлі для busybox

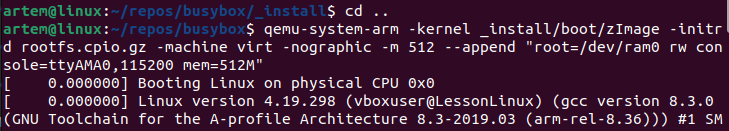


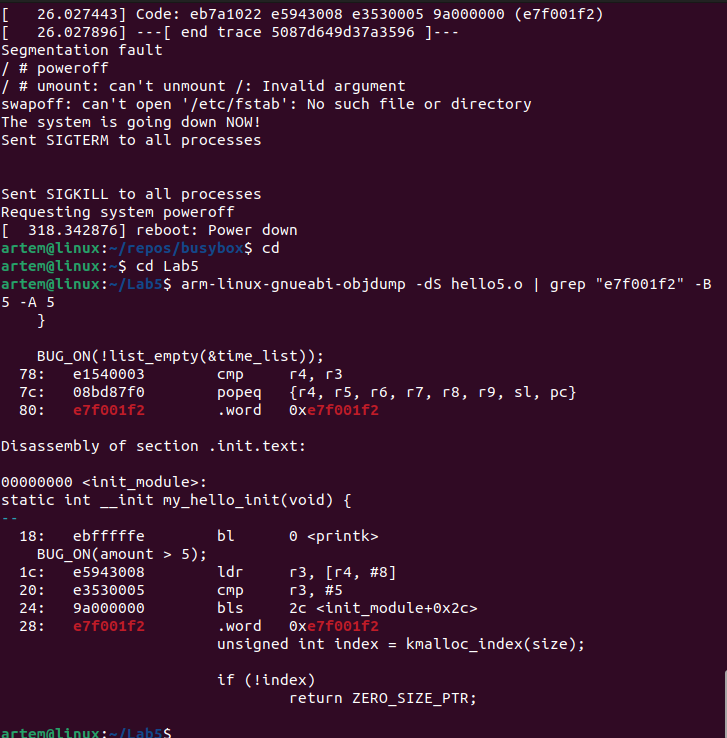


Оновлення busybox



Тест QEMU





**Hello5.c**

#include <linux/init.h>

#include <linux/module.h>

#include <linux/moduleparam.h>

#include <linux/printk.h>

#include <linux/types.h>

#include <linux/ktime.h>

#include <linux/slab.h>

#include <linux/list.h>

struct time\_measurement {

struct list\_head list;

ktime\_t start;

ktime\_t end;

};

MODULE\_AUTHOR("IO-16 Artem");

MODULE\_DESCRIPTION("Lab5");

MODULE\_LICENSE("GPL");

static LIST\_HEAD(time\_list);

static uint amount = 1;

module\_param(amount, uint, 0444);

MODULE\_PARM\_DESC(amount, "An amount parameter");

static int \_\_init my\_hello\_init(void);

static void \_\_exit my\_hello\_exit(void);

static int \_\_init my\_hello\_init(void) {

uint i = 0;

struct time\_measurement \*current\_record;

pr\_info("Amount: %d\n", amount);

BUG\_ON(amount > 5);

for (i = 0; i < amount; i++) {

current\_record = kmalloc(sizeof(\*current\_record), GFP\_KERNEL);

if (!current\_record)

goto error\_handle;

current\_record->start = ktime\_get();

pr\_info("Hello, Maria!\n");

current\_record->end = ktime\_get();

list\_add\_tail(&current\_record->list, &time\_list);

}

return 0;

error\_handle:

pr\_err("No memory!\n");

my\_hello\_exit();

return -ENOMEM;

}

static void \_\_exit my\_hello\_exit(void) {

struct time\_measurement \*temp\_record, \*safe\_record;

list\_for\_each\_entry\_safe(temp\_record, safe\_record, &time\_list, list) {

pr\_info("Execution Time: %lld ns",

ktime\_to\_ns(temp\_record->end - temp\_record->start));

list\_del(&temp\_record->list);

kfree(temp\_record);

}

BUG\_ON(!list\_empty(&time\_list));

}

module\_init(my\_hello\_init);

module\_exit(my\_hello\_exit);

**Висновок:**

Замість виведення повідомлення та повернення -EINVAL для неприпустимих значень параметра, у коді використано макрос BUG\_ON(). Цей макрос призводить до аварійного зупину ядра, якщо значення параметра виходить за допустимі межі. Також до коду було додано спеціальну умову, що імітує помилку виділення пам'яті через kmalloc(), яка умовно повертає нуль. Зроблені скріншоти демонструють коректну роботу програми з урахуванням внесених змін.