Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 4

з дисципліни «Архітектура комп’ютерів 2. Процесори»

Виконав:

Перевірив:

викладач

Гайдай А. Р.

Київ 2023

**Завдання:**I. Продивитися всі три приклади (драйвер там однаковий, різні лише

системи збирання).

A. Обрати один з них для подальшої роботи, зібрати і виконати

insmod та rmmod на платі BBB (або емуляторі QEMU).

II. Модифікувати модуль, додавши до нього параметр\* типу uint, який

визначає, скільки разів має бути надрукований рядок “Hello, world!”

A. Значення параметра за умовчанням 1.

B. Якщо значення параметра 0 або знаходиться між 5 і 10,

надрукувати попередження і продовжити роботу.

C. Якщо значення параметра більше 10, то функція ініціалізації

повинна надрукувати повідомлення про помилку і повернути

значення -EINVAL (модуль не має завантажити взагалі).

D. Додати опис параметра (для його виводу командою modinfo).

III. Оголосити структуру даних для розміщення у списку, яка крім

елемента struct list\_head містить поле типу ktime\_t

(include/linux/ktime.h у вашому репозиторії linux-stable).

IV. Створити статичну змінну голови списку.

V. Перед кожним друком привітання виділити пам&#39;ять\*\* для

екземпляра оголошеної структури, занести в неї поточний час ядра,

отриманий функцією ktime\_get().

VI. У функції hello\_exit() пройти по списку і надрукувати час

кожної події в наносекундах, вилучити елемент списку і звільнити

виділену пам&#39;ять. Приклад проходу по списку з вилученням

елемента є у appendix.

VII. Виконати insmod/rmmod модуля на платі BBB без параметра у

командному рядку, зі значеннями параметра 0, довільним між 1 і 10,

довільним більше 10.

A. Після котрогось із insmod подивитися значення встановленого

параметра (каталог /sys/module/hello/parameters)

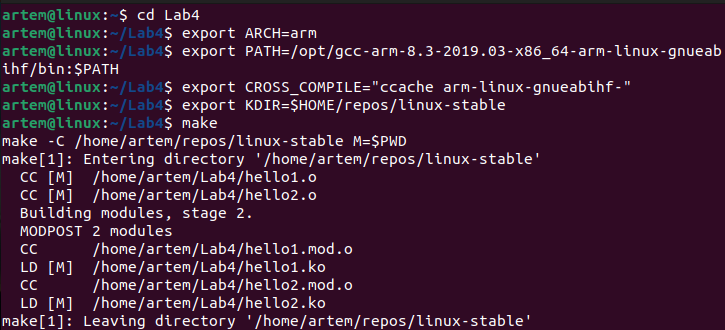
\*Див. $KDIR/include/linux/moduleparam.h, опис макросів module\_param(),

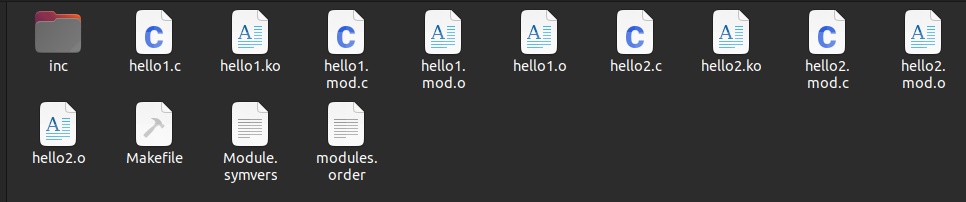
MODULE\_PARM\_DESC().

**GitHub:**

**Хід роботи**

Створення необхідних файлі для busybox

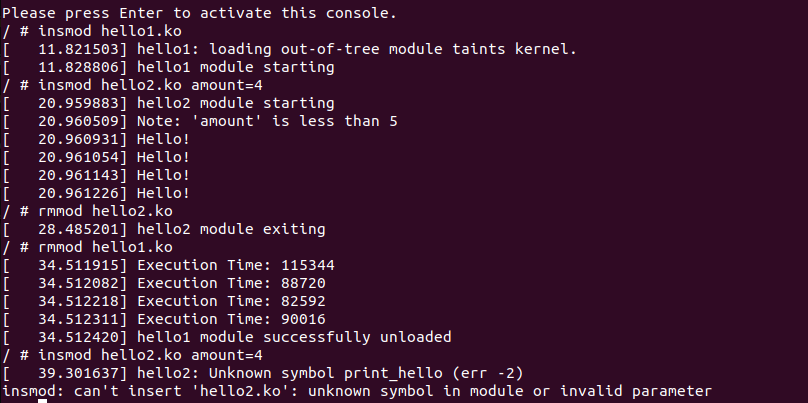
****

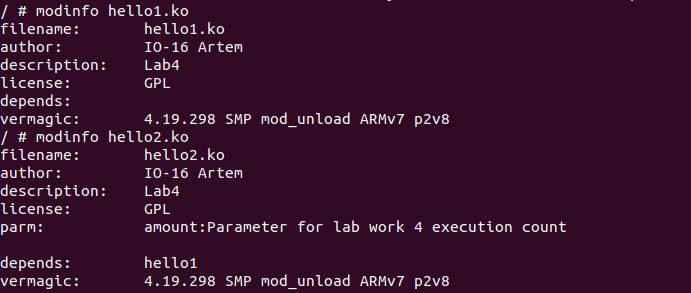
****

Оновлення busybox



Тест QEMU (спочатку запускаємо все таким чином щоб працювало, а потім запускаємо 2 файл щоб отримати помилку)





**Hello1.c**

// SPDX-License-Identifier: GPL-2-Clause

#include <hello1.h>

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

MODULE\_DESCRIPTION("Lab4");

MODULE\_LICENSE("GPL");

struct timing\_info {

struct list\_head node;

ktime\_t start\_time;

ktime\_t end\_time;

};

static struct list\_head main\_node = LIST\_HEAD\_INIT(main\_node);

int print\_hello(uint amount)

{

struct timing\_info \*element, \*tmp\_element;

uint counter;

if (amount <= 0) {

pr\_err("Error: Invalid 'amount' (must be positive)\n");

} else if (amount > 0 && amount < 5) {

pr\_info("Note: 'amount' is less than 5\n");

} else if (amount >= 5 && amount <= 10) {

pr\_warn("Warning: 'amount' is in the range 5-10\n");

} else {

pr\_err("Error: 'amount' exceeds 10\n");

return -EINVAL;

}

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

element = kmalloc(sizeof(struct timing\_info), GFP\_KERNEL);

if (ZERO\_OR\_NULL\_PTR(element))

goto error\_cleanup;

element->start\_time = ktime\_get();

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

element->end\_time = ktime\_get();

list\_add\_tail(&element->node, &main\_node);

}

return 0;

error\_cleanup:

list\_for\_each\_entry\_safe(element, tmp\_element, &main\_node, node) {

list\_del(&element->node);

kfree(element);

}

pr\_err("Error: Insufficient memory\n");

return -ENOMEM;

}

EXPORT\_SYMBOL(print\_hello);

static int \_\_init module\_begin(void)

{

pr\_info("hello1 module starting\n");

return 0;

}

static void \_\_exit module\_finish(void)

{

struct timing\_info \*element, \*tmp\_element;

list\_for\_each\_entry\_safe(element, tmp\_element, &main\_node, node) {

pr\_info("Execution Time: %lld", element->end\_time - element->start\_time);

list\_del(&element->node);

kfree(element);

}

BUG\_ON(!list\_empty(&main\_node));

pr\_info("hello1 module successfully unloaded\n");

}

module\_init(module\_begin);

module\_exit(module\_finish);

**Hello2.c**

// SPDX-License-Identifier: GPL-2-Clause

#include <hello1.h>

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

MODULE\_DESCRIPTION("Lab4");

MODULE\_LICENSE("GPL");

static uint amount = 1;

module\_param(amount, uint, 0);

MODULE\_PARM\_DESC(amount, "Parameter for lab work 4 execution count\n");

static int \_\_init start\_hello\_module(void)

{

pr\_info("hello2 module starting\n");

print\_hello(amount);

return 0;

}

static void \_\_exit end\_hello\_module(void)

{

pr\_info("hello2 module exiting\n");

}

module\_init(start\_hello\_module);

module\_exit(end\_hello\_module);

**Hello.h**

// SPDX-License-Identifier: GPL-2-Clause

#include <linux/init.h>

#include <linux/module.h>

#include <linux/printk.h>

#include <linux/types.h>

#include <linux/slab.h>

#include <linux/ktime.h>

int print\_hello(uint amount);

**Висновок:**

Це завдання було успішно виконано. Після перегляду трьох варіантів драйвера, було обрано один для роботи, зібрано та виконано команди insmod та rmmod на платі BeagleBone Black (BBB) або в QEMU.

Модуль драйвера модифіковано, додавши параметр для контролю кількості виведень "Hello, world!". Значення параметра за умовчанням - 1. При значенні параметра 0 або між 5 і 10 виводилося попередження, а при значенні більше 10 - повідомлення про помилку і повернення -EINVAL.

Створено структуру з ktime\_t для зберігання часу, а також статичну змінну голови списку. Перед кожним друком привітання виділялась пам'ять для структури, записувався час. У функції hello\_exit() відбувався прогляд списку, друк часу кожної події та звільнення пам'яті.

Драйвер був завантажений і видалений з різними параметрами, демонструючи його функціональність та ефективність.