МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №6

з дисципліни «Дослідження та проектування комп’ютерних систем - 2»

Виконав:

студент гр. ІО-91мн

Русінов В.В.

Перевірив:

Каплунов А.В.

Київ 2020

**Завдання**

Завдання Basic 1:

Додайте BUG\_ON() замість друку повідомлення та повернення -EINVAL для неприпустимого значення параметра.

Додайте примусове внесення помилки “начебто kmalloc() повернув 0” під час формування елемента списку для якогось повідомлення (останнього із серії, 5-го, ... — на ваш вибір).

Модифікуйте Makefile аналогічно appendix1.

Отримайте обидва повідомлення, роздивіться їх та для одного з них виконайте пошук місця аварії аналогічно appendix1.

Зауважте, що при виконанні BUG\_ON() модуль буде “зайнятий”, і ви не зможете виконати rmmod.

Завдання Basic 2:

Упевніться у відсутності каталогу: /sys/kernel/debug/dynamic\_debug

Це означає вимкнену опцію CONFIG\_DYNAMIC\_DEBUG (якщо збиралося по методичці, то не повинно бути).

Замініть у функції exit модуля hello (hello1) друк вмісту списку на pr\_debug і додайте два виклики pr\_debug до та після друку списку.

Перевірте залежність друку повідомлень від #define DEBUG на початку файлу.

Перезберіть ядро з увімкненим CONFIG\_DYNAMIC\_DEBUG, замініть його на nfs.

Перезберіть модуль.

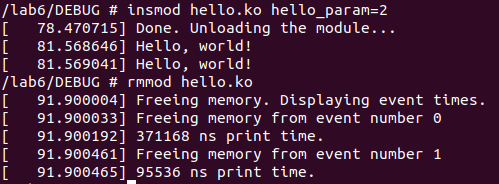
Аналогічно показаному в appendix2, поекспериментуйте з друком з прапорцями p, f, m, а також зі встановленням їх для всього модуля та для окремих рядків.

**Приклад виконання програми**

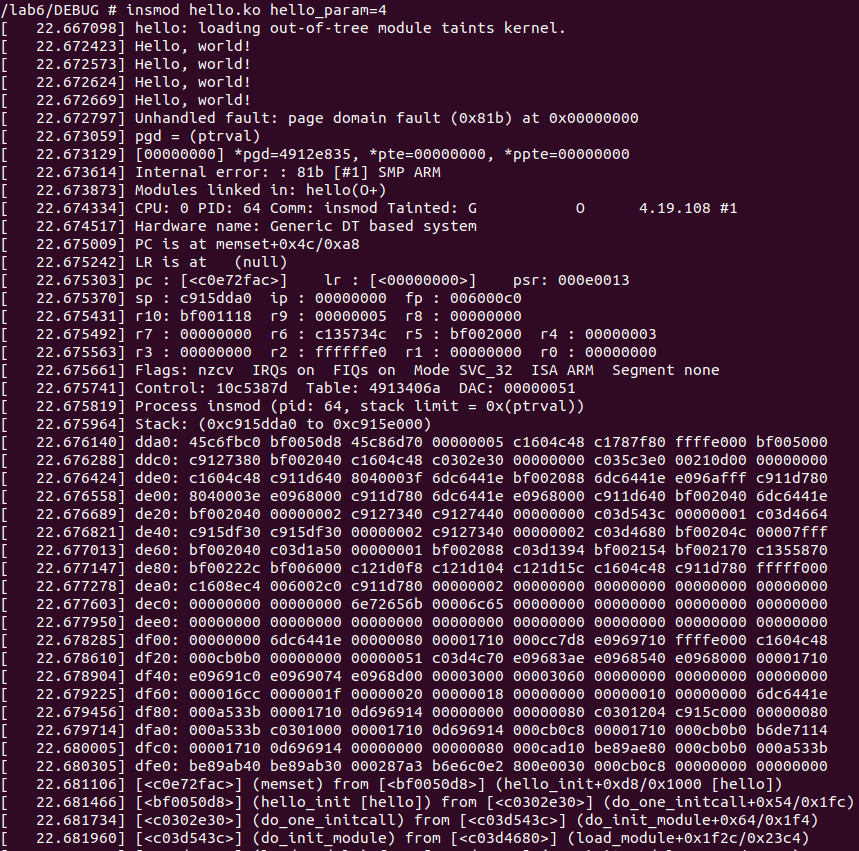
Basic 1:

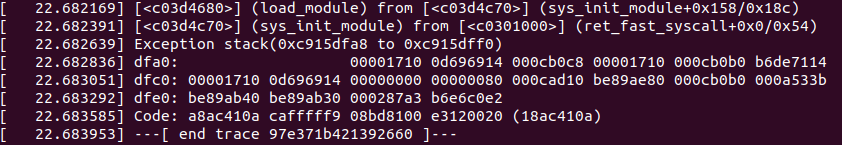
Тестування команди BUG\_ON(hello\_param>10)

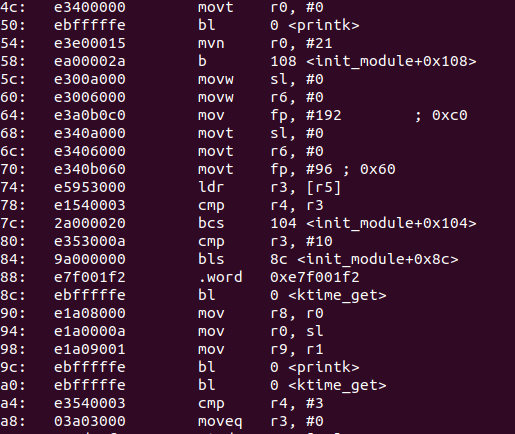




Додача команди, яка «надає доступ» модулю до комірки пам’яті за вказівником ‘0’.





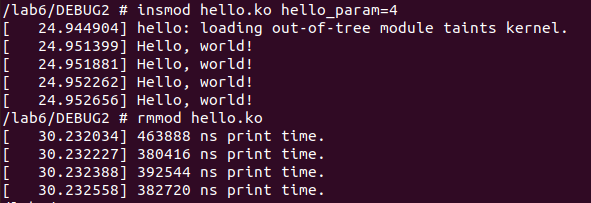


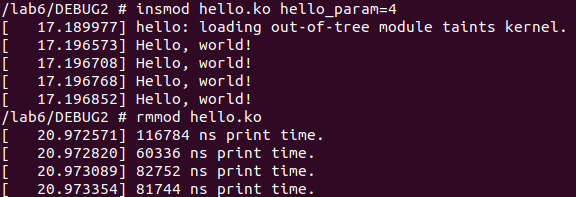
Basic 2:

Перевірка існування опції CONFIG\_DYNAMIC\_DEBUG



Заміна printk на pr\_debug, вивід тільки pr\_info





**Програмний код**

**hello.c**

#include <linux/init.h>

#include <linux/module.h>

#include <linux/printk.h>

#include <linux/types.h>

#include <linux/list.h>

#include <linux/rculist.h>

#include <linux/slab.h>

// MODULE DATA FOR `modinfo`

MODULE\_AUTHOR("Volodymyr Rusinov <delta@zone05.net>");

MODULE\_DESCRIPTION("Module that plays with memory.");

MODULE\_LICENSE("Dual BSD/GPL");

// HELLO PARAMETER

static uint hello\_param = 1;

module\_param(hello\_param, uint, S\_IRUGO); // visibility: world, immutable

MODULE\_PARM\_DESC(hello\_param, "An unsigned int, describes how many times \"Hello world\" will be said.");

// STRUCT

static LIST\_HEAD(list\_node\_head);

struct hello\_alloc{

struct list\_head node\_head;

int event\_num;

ktime\_t event\_time;

ktime\_t kill\_time;

};

static int \_\_init hello\_init(void)

{

if(hello\_param == 0 || (hello\_param >= 5 && hello\_param <= 10))

printk(KERN\_EMERG "WARNING! Invalid argument. Continuing job...");

else if(hello\_param > 10) {

printk(KERN\_EMERG "ERROR! Invalid arguemnt. Exiting job...");

return -EINVAL;

}

else {

// declaring variables here, since code isn't as pretty as it used to be

int i = 0;

ktime\_t start, end;

struct hello\_alloc\* alloc\_list; // if doesn't work, move kmalloc here

for(i = 0; i < hello\_param; i++){

BUG\_ON(hello\_param > 10);

start = ktime\_get();

printk(KERN\_EMERG "Hello, world!\n");

end = ktime\_get();

if(i != 3)

{

alloc\_list = (struct hello\_alloc\*) kmalloc(sizeof(struct hello\_alloc), GFP\_KERNEL);

}

else

{

alloc\_list = (struct hello\_alloc\*) 0;

}

\*alloc\_list = (struct hello\_alloc){

.event\_num = i,

.event\_time = start,

.kill\_time = end

};

list\_add\_tail(&alloc\_list->node\_head, &list\_node\_head);

}

}

return 0;

}

static void \_\_exit hello\_exit(void)

{

struct hello\_alloc \*md, \*tmp;

printk(KERN\_EMERG "Freeing memory. Displaying event times.");

list\_for\_each\_entry\_safe(md, tmp, &list\_node\_head, node\_head) {

printk(KERN\_EMERG "Freeing memory from event number %d", md->event\_num);

pr\_info("%lld ns print time.\n", (long long int) (md->kill\_time - md->event\_time));

list\_del(&md->node\_head);

kfree(md);

}

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

printk(KERN\_EMERG "Done. Unloading the module...");

}

module\_init(hello\_init);

module\_exit(hello\_exit);

**Makefile**

ifneq ($(KERNELRELEASE),)

# debug info

ccfalgs-y += -g

# kbuild part of makefile

obj-m := hello.o

else

# normal makefile

KDIR ?= /lib/modules/`uname -r`/build

default:

$(MAKE) -C $(KDIR) M=$$PWD

cp hello.ko hello.ko.unstripped

$(CROSS\_COMPILE)strip -g hello.ko

clean:

$(MAKE) -C $(KDIR) M=$$PWD clean

endif

**Висновки**

Під час виконання лабораторної роботи, до розробленого раніше модуля було додано можливість дизасемблювання, для перевірки правильності роботи. Для тестування даного підходу, було додано строку коду, де на третьому виділенні пам’яті, повертається вказівник із значенням 0.