МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Архітектура комп’ютерів

Лабораторна робота №4

«Розроблення модулів Linux Kernel (частина 1)»

Виконав:
студент групи ІО-32
Крадожон М.

Перевірив(-ла):
Каплунов А.

2023

Лабораторна робота №4

Мета

Ознайомитися з процесом створення, компіляції, завантаження та видалення модулів ядра Linux у системі зі спільною пам’яттю (на прикладі емулятора QEMU).

Хід роботи

Підготовка середовища

Для виконання роботи використано контейнер Docker, створений у лабораторній роботі №3.
У ньому вже було зібрано ядро Linux для ARM та створено файлову систему rootfs.cpio.gz для QEMU.

Перевірка каталогу ядра:

```
ls /workspace/repos/linux-stable
```

Створення модуля ядра

Було створено два файли:

`hello.c` — вихідний код модуля

`Makefile` — для компіляції модуля під ядро ARM

Файл hello.c

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/errno.h>
#include <linux/slab.h>        // kmalloc, kfree
#include <linux/list.h>        // списки
#include <linux/ktime.h>       // час ядра

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Maxim Kradozhon");
MODULE_DESCRIPTION("Hello world module with list and time tracking");
MODULE_VERSION("1.2");

static uint times = 1;
module_param(times, uint, 0444);
MODULE_PARM_DESC(times, "Кількість виведених повідомлень 'Hello, world!' (uint)");

// структура для зберігання часу виклику
struct hello_item {
    struct list_head list;
    ktime_t timestamp;
};

static LIST_HEAD(hello_list); // статична голова списку

static int __init hello_init(void)
{
    int i;
    struct hello_item *item;

    if (times == 0 || (times >= 5 && times <= 10)) {
        pr_warn("hello: Warning! times=%u — граничне значення, але продовжую\n", times);
    }

    if (times > 10) {
        pr_err("hello: Error! times=%u > 10 — модуль не буде завантажено\n", times);
        return -EINVAL;
    }

    for (i = 0; i < times; i++) {
        item = kmalloc(sizeof(*item), GFP_KERNEL);
        if (!item) {
            pr_err("hello: Помилка виділення пам’яті\n");
            return -ENOMEM;
```

```
        }

        item->timestamp = ktime_get();
        list_add_tail(&item->list, &hello_list);
        pr_info("Hello, world! (%d)\n", i + 1);
    }

    return 0;
}

static void __exit hello_exit(void)
{
    struct hello_item *item, *tmp;

    pr_info("Goodbye, world! Вивід часу подій:\n");

    list_for_each_entry_safe(item, tmp, &hello_list, list) {
        pr_info("  time(ns): %lld\n", ktime_to_ns(item->timestamp));
        list_del(&item->list);
        kfree(item);
    }
}

module_init(hello_init);
module_exit(hello_exit);
```

Makefile

```
obj-m += hello.o

all:
        make -C /workspace/repos/linux-stable M=$(PWD) ARCH=arm CROSS_COMPILE=arm-lin

clean:
        make -C /workspace/repos/linux-stable M=$(PWD) clean
```

Компіляція модуля

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

Результат:

```
root@ebe046d6188a:/workspace# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
make -C /workspace/repos/linux-stable M=/workspace modules
make[1]: Entering directory '/workspace/repos/linux-stable'
  CC [M]  /workspace/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /workspace/hello.mod.o
  LD [M]  /workspace/hello.ko
make[1]: Leaving directory '/workspace/repos/linux-stable'
```

Додавання модуля до rootfs

```
mkdir rootfs
cd rootfs
gunzip -c ../rootfs.cpio.gz | cpio -idmv
cp ../hello.ko ./root/
find . | cpio -o -H newc | gzip > ../rootfs_new.cpio.gz
cd ..
rm -rf rootfs
```

Запуск QEMU

```
cd /workspace/repos/busybox

qemu-system-arm -kernel _install/boot/zImage -initrd rootfs_new.cpio.gz \
-machine virt -nographic -m 512 \
--append "root=/dev/ram0 rw console=ttyAMA0,115200 mem=512M"
```

Перевірка роботи модуля

Усередині QEMU:

```
/ # cd root/
~ # insmod hello.ko times=3
[  122.116754] hello: loading out-of-tree module taints kernel.
[  122.137783] Hello, world! (1)
[  122.138260] Hello, world! (2)
[  122.138601] Hello, world! (3)
~ # rmmod hello
[  131.060258] Goodbye, world! Вивід часу подій:
[  131.061502]   time(ns): 122103593152
[  131.062055]   time(ns): 122104142640
[  131.062446]   time(ns): 122104480672
~ # insmod hello.ko times=1
[  347.497245] Hello, world! (1)
~ # rmmod hello
[  356.176184] Goodbye, world! Вивід часу подій:
[  356.176838]   time(ns): 347463097376
~ # insmod hello.ko times=0
[  385.092817] hello: Warning! times=0 — граничне значення, але продовжую
~ # rmmod hello
[  388.121395] Goodbye, world! Вивід часу подій:
~ # insmod hello.ko times=11
[  410.662621] hello: Error! times=11 > 10 — модуль не буде завантажено
insmod: can`t insert 'hello.ko': invalid parameter
~ # modinfo hello.ko
filename:       hello.ko
author:         Maxim Kradozhon
description:    Hello world module with list and time tracking
license:        GPL
parm:           times:Кількість виведених повідомлень 'Hello, world!' (uint)
version:        1.2
srcversion:     56B14870DDBA2575F83DD22
depends:
vermagic:       4.19.325 SMP mod_unload ARMv7 p2v8
~ # cat /sys/module/hello/parameters/times
3
```

Результати роботи

- Модуль `hello.ko` створено, скомпільовано й завантажено без помилок.
- Параметр `times` успішно передається під час завантаження.
- Роботу підтверджено через `dmesg`, `modinfo` та `/sys/module/hello/parameters/times`.
- Модуль коректно виводить повідомлення при завантаженні та видаленні.

Висновки

У ході виконання роботи:

- розроблено власний модуль ядра Linux для ARM;
- виконано крос-компіляцію та тестування у QEMU;
- вивчено механізм параметрів модулів (`module_param`);
- опановано роботу з інструментами `insmod` , `rmmod` , `modinfo` ;
- отримано практичні навички роботи з модулями ядра Linux.