МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

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

КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Архітектура комп’ютерів

Лабораторна робота №6

«Розроблення модулів Linux Kernel (частина 3)»

Виконав:
студент групи ІО-32
Крадожон М.

Перевірив(-ла):
Каплунов А.

2023

Лабораторна робота №6

Мета

Ознайомитися з процесом створення, завантаження та аварійного тестування модуля ядра Linux.
Навчитися працювати зі списками структур (`list_head`), параметрами модуля (`module_param`) і вимірювати час виконання подій у ядрі (`ktime_t`).
Демонструвати обробку критичних умов за допомогою `BUG_ON` та симуляцію помилки `kmalloc` .

Хід роботи

Підготовка середовища

Для виконання роботи використано контейнер Docker із QEMU (створений у лабораторній роботі №3).
У контейнері вже зібрано ядро ARM і є файлову систему rootfs.cpio.gz.

Перевірка каталогу ядра:

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

Створення модуля ядра

Модуль `hello.ko` виконує наступне:

- Параметр `times` задає кількість повідомлень `"Hello, world!"` .
- При `times == 0` або `5 ≤ times ≤ 10` виводиться попередження (`pr_warn`).
- При `times > 10` виконується `BUG_ON` , що блокує завантаження модуля.
- Кожен виклик створює структуру `hello_item` у списку `hello_list` і зберігає час виклику (`ktime_get()`).
- Симулюється помилка `kmalloc()` для 5-го елемента (`item = NULL`), що викликає kernel OOPS.

Файл hello.c

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/errno.h>
#include <linux/slab.h>
#include <linux/list.h>
#include <linux/ktime.h>
#include <linux/bug.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Maxim Kradozhon");
MODULE_DESCRIPTION("Hello world module with list and time tracking (LR4 -> LR6 task1)");
MODULE_VERSION("1.3");

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

struct hello_item {
    struct list_head list;
    ktime_t timestamp;
};

static LIST_HEAD(hello_list);

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);
    }

    BUG_ON(times > 10);

    for (i = 0; i < times; i++) {
        if (i == 4) {
            pr_warn("hello: Симуляція помилки kmalloc() для елемента %d\n", i);
            item = NULL;
        } else {
            item = kmalloc(sizeof(*item), GFP_KERNEL);
        }
```

```
        BUG_ON(!item);

        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

```
ifneq ($(KERNELRELEASE),)
# kbuild part of makefile
obj-m := hello.o
ccflags-y += -g
else
KDIR ?= /lib/modules/`uname -r`/build
PWD := $(shell pwd)

default:
	$(MAKE) -C $(KDIR) M=$(PWD) modules
	cp hello.ko hello.ko.unstripped
	strip -g hello.ko

clean:
	$(MAKE) -C $(KDIR) M=$(PWD) clean
endif
```

Компіляція модуля

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

Результат:

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

Додавання модуля до rootfs

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

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=11
[   85.763151] hello: loading out-of-tree module taints kernel.
[   85.779149] ------------[ cut here ]------------
[   85.779637] kernel BUG at /workspace/hello/hello.c:35!
[   85.780152] Internal error: Oops - BUG: 0 [#1] SMP ARM
[   85.781007] Modules linked in: hello(O+)
[   85.782595] CPU: 0 PID: 68 Comm: insmod Tainted: G           O      4.19.325 #1
[   85.783227] Hardware name: Generic DT based system
[   85.785223] PC is at hello_init+0x38/0x1000 [hello]
[   85.786233] LR is at do_one_initcall+0x54/0x214
[   85.786735] pc : [<bf005038>]    lr : [<c030362c>]    psr: 200f0013
[   85.787363] sp : c8b45db8  ip : c8afc940  fp : bf002040
[   85.787905] r10: 00000000  r9 : c1704c48  r8 : 00000000
[   85.788467] r7 : bf005000  r6 : ffffe000  r5 : bf002000  r4 : c18881a0
[   85.789135] r3 : 0000000b  r2 : 674d7172  r1 : 0000000b  r0 : 00000000
[   85.789934] Flags: nzCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment none
[   85.790438] Control: 10c5387d  Table: 48b0806a  DAC: 00000051
[   85.791446] Process insmod (pid: 68, stack limit = 0x66532ea9)
[   85.791801] Stack: (0xc8b45db8 to 0xc8b46000)
[   85.792191] 5da0:                                                c18881a0 c17
[   85.793272] 5dc0: ffffe000 bf005000 00000000 c1704c48 00000000 c030362c c8afc840 c04
[   85.794027] 5de0: 00210d00 00000002 c1704c48 c8aaf080 8040003f c04f36c0 00000000 674
[   85.794711] 5e00: ffe00000 c8aaf8c0 8040003e e0968000 c8aaf8c0 674d7172 e0968000 c8a
[   85.795375] 5e20: bf002040 674d7172 bf002040 00000002 c8afc900 00000002 c8afc840 c03
[   85.796044] 5e40: 00000001 c03d4ee4 c8b45f30 c8b45f30 00000002 c8afc800 00000002 c03
[   85.797018] 5e60: bf00204c 00007fff bf002040 c03d1d88 c8af4818 bf002088 c8af4a78 bf0
[   85.797962] 5e80: 00000001 bf002170 c135be18 c1221c34 c1221ca4 c1704c48 c1708f04 c8a
[   85.798673] 5ea0: fffff000 e0800000 c8aaf8c0 c8aaf080 00000000 00000000 00000000 000
[   85.799346] 5ec0: 00000000 00000000 6e72656b 00006c65 00000000 00000000 00000000 000
[   85.800010] 5ee0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000
[   85.801066] 5f00: 00000000 674d7172 00000080 0000bf14 00000000 e0973f14 00127164 c17
[   85.802008] 5f20: 0011b1f8 ffffe000 00000051 c03d5364 e0968432 e09684c0 e0968000 000
[   85.802955] 5f40: e097376c e0973588 e0971008 00003000 00003040 00000000 00000000 000
[   85.803908] 5f60: 00001760 0000002e 0000002f 00000019 00000000 00000012 00000000 674
[   85.804903] 5f80: 000f411e 0011b1f8 b6f64950 0000bf14 00000080 c0301264 c8b44000 000
[   85.805925] 5fa0: 000f411e c0301000 0011b1f8 b6f64950 0011b250 0000bf14 0011b1f8 000
[   85.806648] 5fc0: 0011b1f8 b6f64950 0000bf14 00000080 00000001 bec63e90 001086c4 000
[   85.807314] 5fe0: bec63b48 bec63b38 0003b270 b6e1e1b0 600f0010 0011b250 00000000 000
[   85.809151] [<bf005038>] (hello_init [hello]) from [<c030362c>] (do_one_initcall+0x5
[   85.810354] [<c030362c>] (do_one_initcall) from [<c03d2b4c>] (do_init_module+0x48/0x
[   85.811221] [<c03d2b4c>] (do_init_module) from [<c03d4f00>] (load_module+0x21a8/0x24
[   85.812007] [<c03d4f00>] (load_module) from [<c03d5364>] (sys_init_module+0x158/0x18
[   85.812760] [<c03d5364>] (sys_init_module) from [<c0301000>] (ret_fast_syscall+0x0/0
[   85.813401] Exception stack(0xc8b45fa8 to 0xc8b45ff0)
[   85.813855] 5fa0:                   0011b1f8 b6f64950 0011b250 0000bf14 0011b1f8 000
```

```
[   85.814532] 5fc0: 0011b1f8 b6f64950 0000bf14 00000080 00000001 bec63e90 001086c4 000
[   85.815125] 5fe0: bec63b48 bec63b38 0003b270 b6e1e1b0
[   85.816054] Code: eb7a3ad2 e5953000 e353000a 9a000000 (e7f001f2)
[   85.817545] ---[ end trace c780c704f4b24ffb ]---
Segmentation fault
~ # insmod /root/hello.ko times=5
[  401.219193] hello: loading out-of-tree module taints kernel.
[  401.235124] hello: Warning! times=5 — граничне значення, але продовжую
[  401.236092] Hello, world! (1)
[  401.236422] Hello, world! (2)
[  401.236699] Hello, world! (3)
[  401.236999] Hello, world! (4)
[  401.237314] hello: Симуляція помилки kmalloc() для елемента 4
[  401.239547] ------------[ cut here ]------------
[  401.240075] kernel BUG at /workspace/hello/hello.c:45!
[  401.240658] Internal error: Oops - BUG: 0 [#1] SMP ARM
[  401.241478] Modules linked in: hello(O+)
[  401.242849] CPU: 0 PID: 104 Comm: insmod Tainted: G           O      4.19.325 #1
[  401.243586] Hardware name: Generic DT based system
[  401.245946] PC is at hello_init+0x94/0x1000 [hello]
[  401.246655] LR is at hello_init+0x78/0x1000 [hello]
[  401.247241] pc : [<bf005094>]    lr : [<bf005078>]    psr: 600f0013
[  401.247881] sp : c8b07db8  ip : 00000000  fp : bf002040
[  401.248375] r10: 00000000  r9 : bf002004  r8 : 006000c0
[  401.249032] r7 : c135d34c  r6 : 00000004  r5 : bf002000  r4 : c8af99c0
[  401.249781] r3 : 539f2452  r2 : 539f2452  r1 : 1a569000  r0 : 0000004c
[  401.251122] Flags: nZCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment none
[  401.251888] Control: 10c5387d  Table: 48b0806a  DAC: 00000051
[  401.253629] Process insmod (pid: 104, stack limit = 0x472fc9fc)
[  401.254640] Stack: (0xc8b07db8 to 0xc8b08000)
[  401.255416] 7da0:                                                       c18881a0 c17
[  401.256313] 7dc0: ffffe000 bf005000 00000000 c1704c48 00000000 c030362c c8af9800 c04
[  401.257143] 7de0: 00210d00 00000002 c1704c48 c8ab08c0 8040003f c04f36c0 00000000 539
[  401.257994] 7e00: ffe00000 c8ab0b40 8040003e e0968000 c8ab0b40 539f2452 e0968000 c8a
[  401.258768] 7e20: bf002040 539f2452 bf002040 00000002 c8af98c0 00000002 c8af9800 c03
[  401.259536] 7e40: 00000001 c03d4ee4 c8b07f30 c8b07f30 00000002 c8af97c0 00000002 c03
[  401.260240] 7e60: bf00204c 00007fff bf002040 c03d1d88 c8b26818 bf002088 c8b26a78 bf0
[  401.260928] 7e80: 00000001 bf002170 c135be18 c1221c34 c1221ca4 c1704c48 c1708f04 c8a
[  401.261597] 7ea0: fffff000 e0800000 c8ab0b40 c8ab08c0 00000000 00000000 00000000 000
[  401.262041] 7ec0: 00000000 00000000 6e72656b 00006c65 00000000 00000000 00000000 000
[  401.262752] 7ee0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000
[  401.263791] 7f00: 00000000 539f2452 00000080 0000bf14 00000000 e0973f14 00127164 c17
[  401.264815] 7f20: 0011b1f8 ffffe000 00000051 c03d5364 e0968432 e09684c0 e0968000 000
[  401.265616] 7f40: e097376c e0973588 e0971008 00003000 00003040 00000000 00000000 000
[  401.266288] 7f60: 00001760 0000002e 0000002f 00000019 00000000 00000012 00000000 539
[  401.266719] 7f80: 000f411e 0011b1f8 b6f1c950 0000bf14 00000080 c0301264 c8b06000 000
```

```
[  401.267159] 7fa0: 000f411e c0301000 0011b1f8 b6f1c950 0011b250 0000bf14 0011b1f8 000
[  401.267605] 7fc0: 0011b1f8 b6f1c950 0000bf14 00000080 00000001 bef8de80 001086c4 000
[  401.268024] 7fe0: bef8db38 bef8db28 0003b270 b6dd61b0 600f0010 0011b250 00000000 000
[  401.269764] [<bf005094>] (hello_init [hello]) from [<c030362c>] (do_one_initcall+0x5
[  401.270627] [<c030362c>] (do_one_initcall) from [<c03d2b4c>] (do_init_module+0x48/0x
[  401.271607] [<c03d2b4c>] (do_init_module) from [<c03d4f00>] (load_module+0x21a8/0x24
[  401.272476] [<c03d4f00>] (load_module) from [<c03d5364>] (sys_init_module+0x158/0x18
[  401.273307] [<c03d5364>] (sys_init_module) from [<c0301000>] (ret_fast_syscall+0x0/0
[  401.274221] Exception stack(0xc8b07fa8 to 0xc8b07ff0)
[  401.274897] 7fa0:                   0011b1f8 b6f1c950 0011b250 0000bf14 0011b1f8 000
[  401.275891] 7fc0: 0011b1f8 b6f1c950 0000bf14 00000080 00000001 bef8de80 001086c4 000
[  401.276762] 7fe0: bef8db38 bef8db28 0003b270 b6dd61b0
[  401.277932] Code: e3a02010 eb51b663 e2504000 1a000000 (e7f001f2)
[  401.279252] ---[ end trace 7607ba40560f56ff ]---
Segmentation fault
```

Результати роботи

- Модуль `hello.ko` скомпільовано та завантажено під ARM у QEMU.
- Параметр `times` працює та генерує попередження при граничних значеннях.
- Симуляція `kmalloc` викликала аварію ядра, як передбачалося.
- Час кожного виклику зафіксовано у списку `hello_list`.

Висновки

- Опановано створення і завантаження модуля ядра та роботу з параметрами модуля.
- Демонстровано використання списків і відстеження часу подій у ядрі.
- Підтверджено дію `BUG_ON` та поведінку ядра при критичних помилках.
- Отримано практичні навички роботи з командами `insmod`, `rmmod` та QEMU.

GitHub

GitHub