

Міністерство освіти та науки України
Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

Лабораторна робота №4

з дисципліни “Технології програмування для комп'ютерних систем – 3”

Виконав:
студент групи ІВ-91мн
Гончаренко Олександр

Київ 2020 р.

Завдання

Розглянути етапи створення та використання модулів для ядра OS Linux.

Зауваження щодо збирання модулів:

I. Ядро лінукса вже має бути зібране, щоб були готові всі необхідні для компіляції модулів файли (*робота #3*).

II. Перед виконанням `make`, до команд `export`, які використовувалися для збирання ядра, слід додати `export KDIR=шлях_до_каталогу_ядра` (наприклад, `$HOME/repos/linux-stable`)

III. Для довідки:

A. Каталог `$KDIR/Documentation/kbuild`, зокрема `modules.txt`

B. `$KDIR/Documentation/process/coding-style.rst`

C. Параметри модуля описані в додатку *Building and Running Modules* у *Module Parameters* (сторінка 35)

Завдання Basic:

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!”

. Значення параметра за умовчанням 1.

A. Якщо значення параметра 0 або знаходиться між 5 і 10, надрукувати попередження і продовжити роботу.

B. Якщо значення параметра більше 10, то функція ініціалізації повинна надрукувати повідомлення про помилку і повернути значення `-EINVAL` (модуль не має завантажити взагалі).

III. Додати опис параметра. Подивитися його командою `modinfo`.

IV. Виконати `insmod/rmmod` модуля на платі BBB без параметра у командному рядку, зі значеннями параметра 0, довільним між 1 і 10, довільним більше 10.

. 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()`.

Завдання Advanced:

I. Розділити проект на два модулі, `hello1` та `hello2`.

A. Модуль `hello1` повинен експортувати функцію `print_hello()`, яку використовуватиме модуль `hello2` (параметр кількості викликів функції перенести у модуль `hello2`).

II. Заголовковий файл `hello1.h`, який використовуватимуть обидва модулі, винести у підкаталог `inc`, який додати у систему збирання так, щоб файли `*.c` могли використовувати директиву `#include "hello1.h"` лише з іменем файлу, без шляху (див. `ccflags-y`).

III. Замінити `printk` на відповідні ситуації `pr_err`, `pr_warn`, `pr_info` (для друку привітання використати `pr_info`)

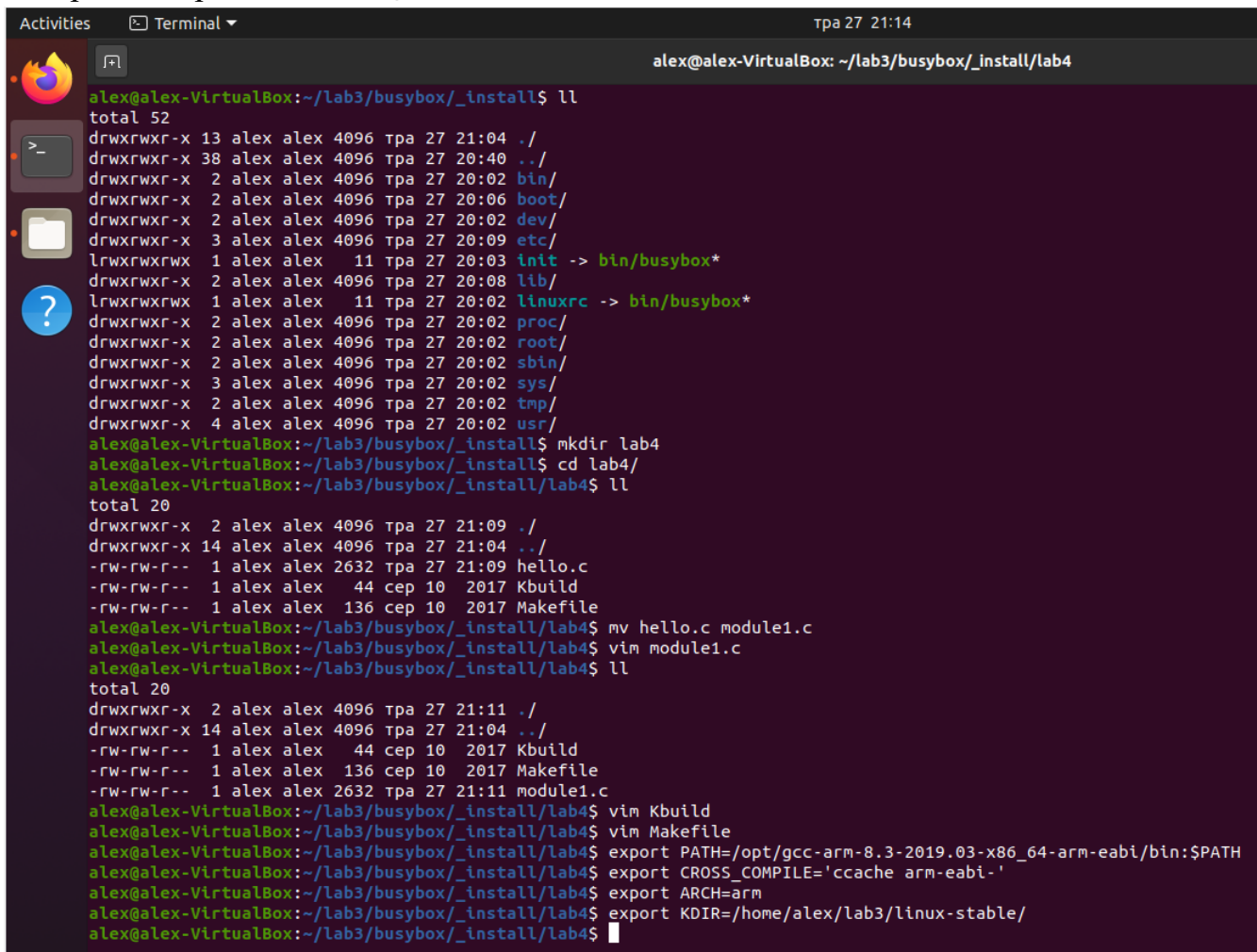
IV. Виконати `insmod hello1.ko`, потім `insmod hello2.ko` з такими значеннями параметра, щоб отримати всі можливі повідомлення і знайти їх (може знадобитися `dmesg`, `grep`).

V. Спробувати завантажити `hello2.ko`, не завантажуючи `hello1.ko`, пояснити результат.

Див. `$KDIR/include/linux/printk.h`

Послідовність виконання роботи

1. Створюємо модуль. Для цього у файловій системі створюємо директорію `lab4`, у яку додаємо усі необхідні файли для роботи з модулем. Виконуємо команди `export` для збирання ядра, а також `export KDIR=/home/alex/lab3/linux-stable/`.



```
alex@alex-VirtualBox: ~/lab3/busybox/_install/lab4
alex@alex-VirtualBox:~/lab3/busybox/_install$ ll
total 52
drwxrwxr-x 13 alex alex 4096 тра 27 21:04 ./
drwxrwxr-x 38 alex alex 4096 тра 27 20:40 ../
drwxrwxr-x 2 alex alex 4096 тра 27 20:02 bin/
drwxrwxr-x 2 alex alex 4096 тра 27 20:06 boot/
drwxrwxr-x 2 alex alex 4096 тра 27 20:02 dev/
drwxrwxr-x 3 alex alex 4096 тра 27 20:09 etc/
lrwxrwxrwx 1 alex alex 11 тра 27 20:03 init -> bin/busybox*
drwxrwxr-x 2 alex alex 4096 тра 27 20:08 lib/
lrwxrwxrwx 1 alex alex 11 тра 27 20:02 linuxrc -> bin/busybox*
drwxrwxr-x 2 alex alex 4096 тра 27 20:02 proc/
drwxrwxr-x 2 alex alex 4096 тра 27 20:02 root/
drwxrwxr-x 2 alex alex 4096 тра 27 20:02 sbin/
drwxrwxr-x 3 alex alex 4096 тра 27 20:02 sys/
drwxrwxr-x 2 alex alex 4096 тра 27 20:02 tmp/
drwxrwxr-x 4 alex alex 4096 тра 27 20:02 usr/
alex@alex-VirtualBox:~/lab3/busybox/_install$ mkdir lab4
alex@alex-VirtualBox:~/lab3/busybox/_install$ cd lab4/
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ ll
total 20
drwxrwxr-x 2 alex alex 4096 тра 27 21:09 ./
drwxrwxr-x 14 alex alex 4096 тра 27 21:04 ../
-rw-rw-r-- 1 alex alex 2632 тра 27 21:09 hello.c
-rw-rw-r-- 1 alex alex 44 сер 10 2017 Kbuild
-rw-rw-r-- 1 alex alex 136 сер 10 2017 Makefile
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ mv hello.c module1.c
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ vim module1.c
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ ll
total 20
drwxrwxr-x 2 alex alex 4096 тра 27 21:11 ./
drwxrwxr-x 14 alex alex 4096 тра 27 21:04 ../
-rw-rw-r-- 1 alex alex 44 сер 10 2017 Kbuild
-rw-rw-r-- 1 alex alex 136 сер 10 2017 Makefile
-rw-rw-r-- 1 alex alex 2632 тра 27 21:11 module1.c
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ vim Kbuild
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ vim Makefile
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ export PATH=/opt/gcc-arm-8.3-2019.03-x86_64-arm-eabi/bin:$PATH
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ export CROSS_COMPILE='ccache arm-eabi-'
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ export ARCH=arm
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ export KDIR=/home/alex/lab3/linux-stable/
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$
```

2. Виконуємо команду make для збирання модулю:

```
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ make
make -C /home/alex/lab3/linux-stable/ M=$PWD
make[1]: Entering directory '/home/alex/lab3/linux-stable'
CC [M] /home/alex/lab3/busybox/_install/lab4/module1.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/alex/lab3/busybox/_install/lab4/module1.mod.o
LD [M] /home/alex/lab3/busybox/_install/lab4/module1.ko
make[1]: Leaving directory '/home/alex/lab3/linux-stable'
```

3. Створимо архів СРІО для rootfs та заархівуємо його за допомогою GZip:

```
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ cd ..
alex@alex-VirtualBox:~/lab3/busybox/_install$ find . | cpio -o -H newc | gzip > ../rootfs.cpio.gz
89642 blocks
alex@alex-VirtualBox:~/lab3/busybox/_install$ cd ..
alex@alex-VirtualBox:~/lab3/busybox$ qemu-system-arm -kernel _install/boot/zImage -initrd rootfs.cpio.gz -machine virt -nographic -m 512 --append "root=/dev/ram0 rw console=ttyAMA0,115200 mem=512M"
```

4. Виконання завдання Basic (module1.c):



```
static unsigned int repeats = 1;

module_param(repeats, uint, S_IRUGO);
MODULE_PARM_DESC(repeats, "Hello print:");

static int __init module1_init(void)
{
    unsigned int repeat;

    if (repeats > 10)
    {
        printk(KERN_ERR "Cannot repeat more than 10 times\n");
        return -EINVAL;
    }

    if (repeats >= 5 && repeats <= 10)
    {
        printk(KERN_WARNING "Repeatition from 5 to 10 times\n");
    }

    if (repeats == 0)
    {
        printk(KERN_WARNING "No repeatition\n");
    }

    for (repeat = 0; repeat < repeats; repeat++)
    {
        printk(KERN_INFO "Hello, world!\n");
    }

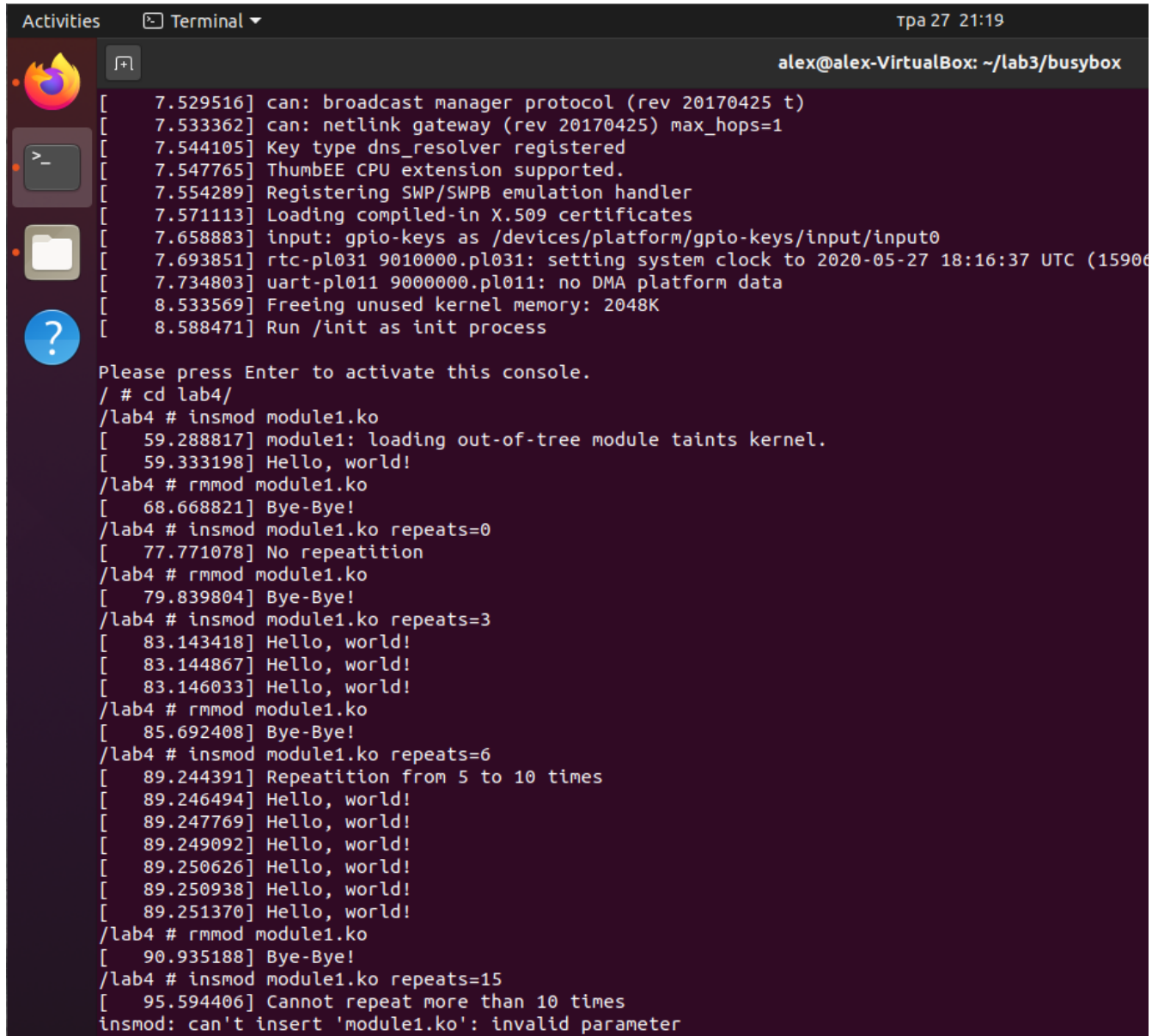
    return 0;
}

static void __exit module1_exit(void)
{
    printk(KERN_EMERG "Bye-Bye!\n");
}

module_init(module1_init);
module_exit(module1_exit);

MODULE_AUTHOR("AlexHoncharenko");
MODULE_DESCRIPTION("Training to build linux module");
MODULE_LICENSE("Dual BSD/GPL");
```

5. Протестуємо роботу модуля:



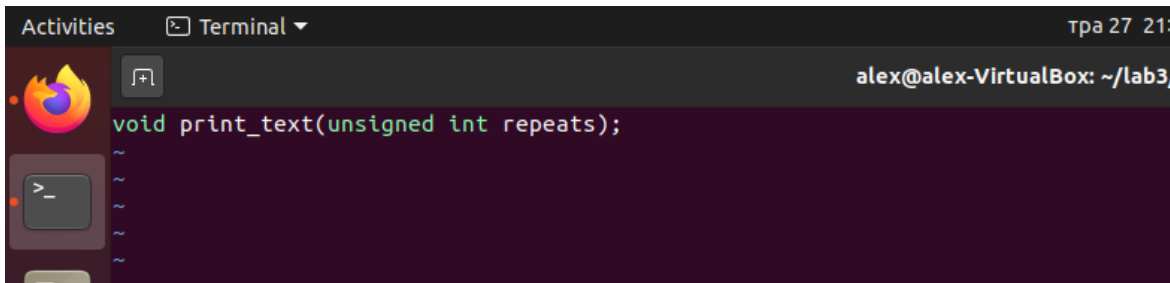
The screenshot shows a terminal window titled "alex@alex-VirtualBox: ~/lab3/busybox". The terminal displays the following content:

```
[ 7.529516] can: broadcast manager protocol (rev 20170425 t)
[ 7.533362] can: netlink gateway (rev 20170425) max_hops=1
[ 7.544105] Key type dns_resolver registered
[ 7.547765] ThumbEE CPU extension supported.
[ 7.554289] Registering SWP/SWPB emulation handler
[ 7.571113] Loading compiled-in X.509 certificates
[ 7.658883] input: gpio-keys as /devices/platform/gpio-keys/input/input0
[ 7.693851] rtc-pl031 9010000.pl031: setting system clock to 2020-05-27 18:16:37 UTC (15906
[ 7.734803] uart-pl011 9000000.pl011: no DMA platform data
[ 8.533569] Freeing unused kernel memory: 2048K
[ 8.588471] Run /init as init process

Please press Enter to activate this console.
/ # cd lab4/
/lab4 # insmod module1.ko
[ 59.288817] module1: loading out-of-tree module taints kernel.
[ 59.333198] Hello, world!
/lab4 # rmmod module1.ko
[ 68.668821] Bye-Bye!
/lab4 # insmod module1.ko repeats=0
[ 77.771078] No repetition
/lab4 # rmmod module1.ko
[ 79.839804] Bye-Bye!
/lab4 # insmod module1.ko repeats=3
[ 83.143418] Hello, world!
[ 83.144867] Hello, world!
[ 83.146033] Hello, world!
/lab4 # rmmod module1.ko
[ 85.692408] Bye-Bye!
/lab4 # insmod module1.ko repeats=6
[ 89.244391] Repeation from 5 to 10 times
[ 89.246494] Hello, world!
[ 89.247769] Hello, world!
[ 89.249092] Hello, world!
[ 89.250626] Hello, world!
[ 89.250938] Hello, world!
[ 89.251370] Hello, world!
/lab4 # rmmod module1.ko
[ 90.935188] Bye-Bye!
/lab4 # insmod module1.ko repeats=15
[ 95.594406] Cannot repeat more than 10 times
insmod: can't insert 'module1.ko': invalid parameter
```

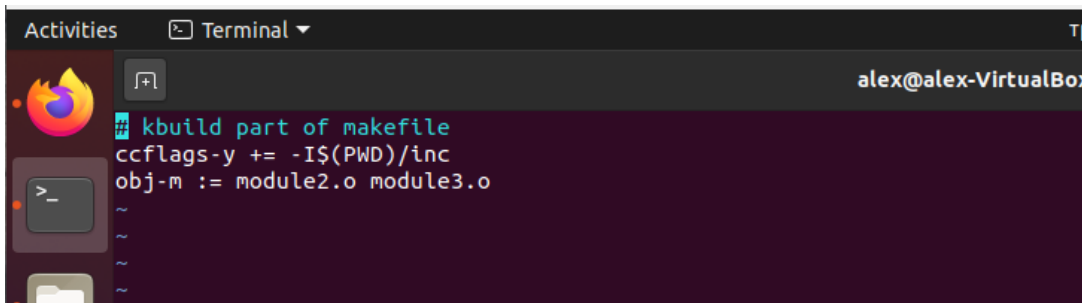
6. Виконання завдання Advanced:

module2.h

A terminal window titled 'alex@alex-VirtualBox: ~/lab3/' showing the content of a file named 'module2.h'. The file contains a function declaration: 'void print_text(unsigned int repeats);' followed by three tilde characters (~) on separate lines.

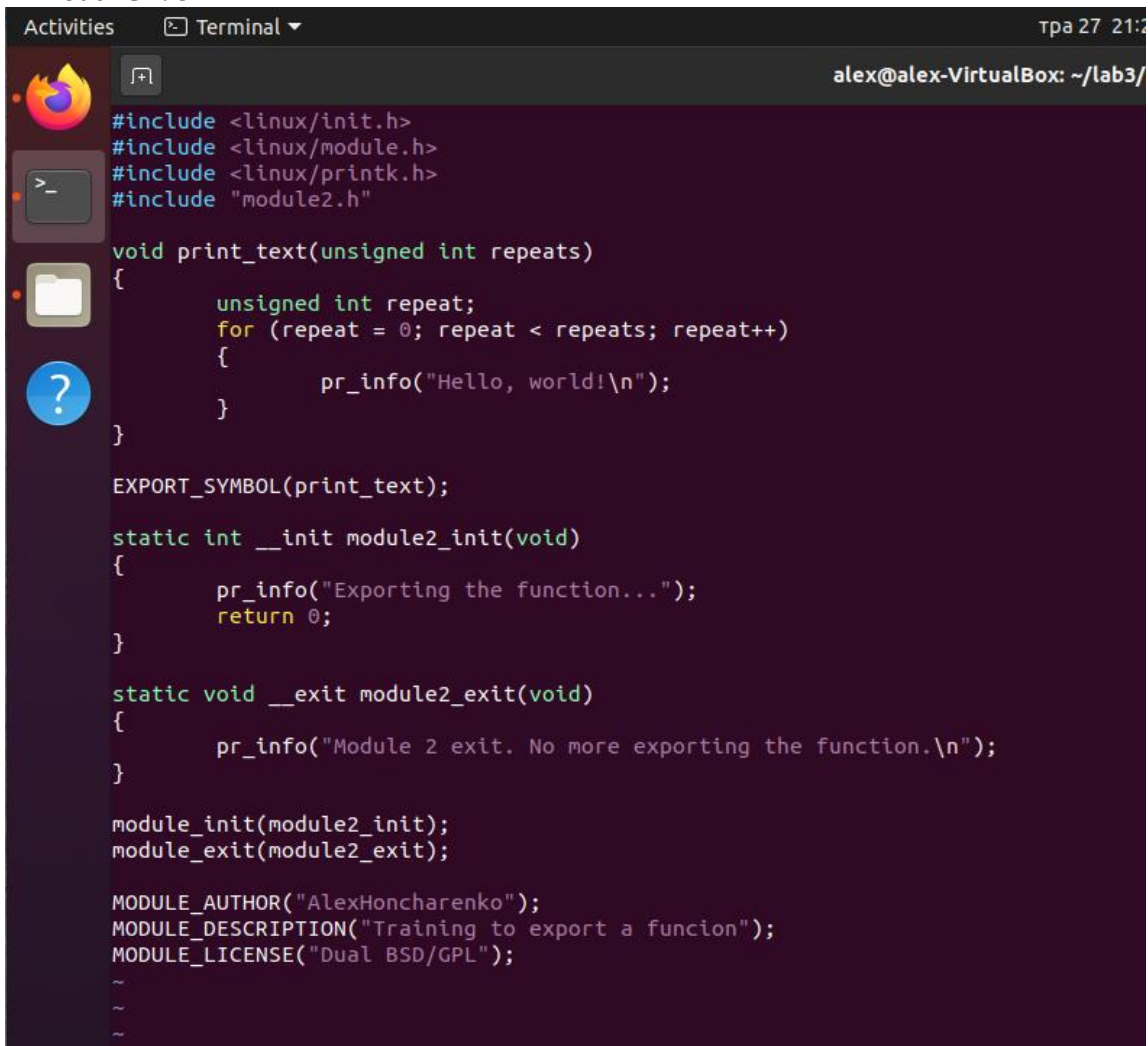
```
void print_text(unsigned int repeats);
~
~
~
```

Kbuild

A terminal window titled 'alex@alex-VirtualBox: ~/lab3/' showing the content of a file named 'Kbuild'. The file contains two lines: '# kbuild part of makefile' and 'ccflags-y += -I\$(PWD)/inc', followed by 'obj-m := module2.o module3.o' and three tilde characters (~) on separate lines.

```
# kbuild part of makefile
ccflags-y += -I$(PWD)/inc
obj-m := module2.o module3.o
~
~
~
```

module2.c

A terminal window titled 'alex@alex-VirtualBox: ~/lab3/' showing the content of a file named 'module2.c'. The file contains C code for a kernel module. It includes headers for linux/init.h, linux/module.h, and linux/printk.h, and includes 'module2.h'. The code defines a 'print_text' function that prints 'Hello, world!' a specified number of times. It also defines '__init' and '__exit' functions for module initialization and cleanup, and registers them with 'module_init' and 'module_exit'. Finally, it sets module metadata like author, description, and license.

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/printk.h>
#include "module2.h"

void print_text(unsigned int repeats)
{
    unsigned int repeat;
    for (repeat = 0; repeat < repeats; repeat++)
    {
        pr_info("Hello, world!\n");
    }
}

EXPORT_SYMBOL(print_text);

static int __init module2_init(void)
{
    pr_info("Exporting the function...");
    return 0;
}

static void __exit module2_exit(void)
{
    pr_info("Module 2 exit. No more exporting the function.\n");
}

module_init(module2_init);
module_exit(module2_exit);

MODULE_AUTHOR("AlexHoncharenko");
MODULE_DESCRIPTION("Training to export a function");
MODULE_LICENSE("Dual BSD/GPL");
~
~
~
```

module3.c

```
Activities Terminal tpa 27 21:28 alex@alex-VirtualBox: ~/lab3/bu

#include <linux/init.h>
#include <linux/module.h>
#include <linux/printk.h>
#include "module2.h"

static unsigned int repeats = 1;

module_param(repeats, uint, S_IRUGO);
MODULE_PARM_DESC(repeats, "Hello print:");

static int __init module3_init(void)
{
    if (repeats > 10)
    {
        pr_err("Cannot repeat more than 10 times\n");
        return -EINVAL;
    }

    if (repeats >= 5 && repeats <= 10)
    {
        pr_warn("Repeattition from 5 to 10 times\n");
    }

    if (repeats == 0)
    {
        pr_warn("No repeattition\n");
    }

    print_text(repeats);
    return 0;
}

static void __exit module3_exit(void)
{
    pr_info("Bye-Bye!\n");
}

module_init(module3_init);
module_exit(module3_exit);

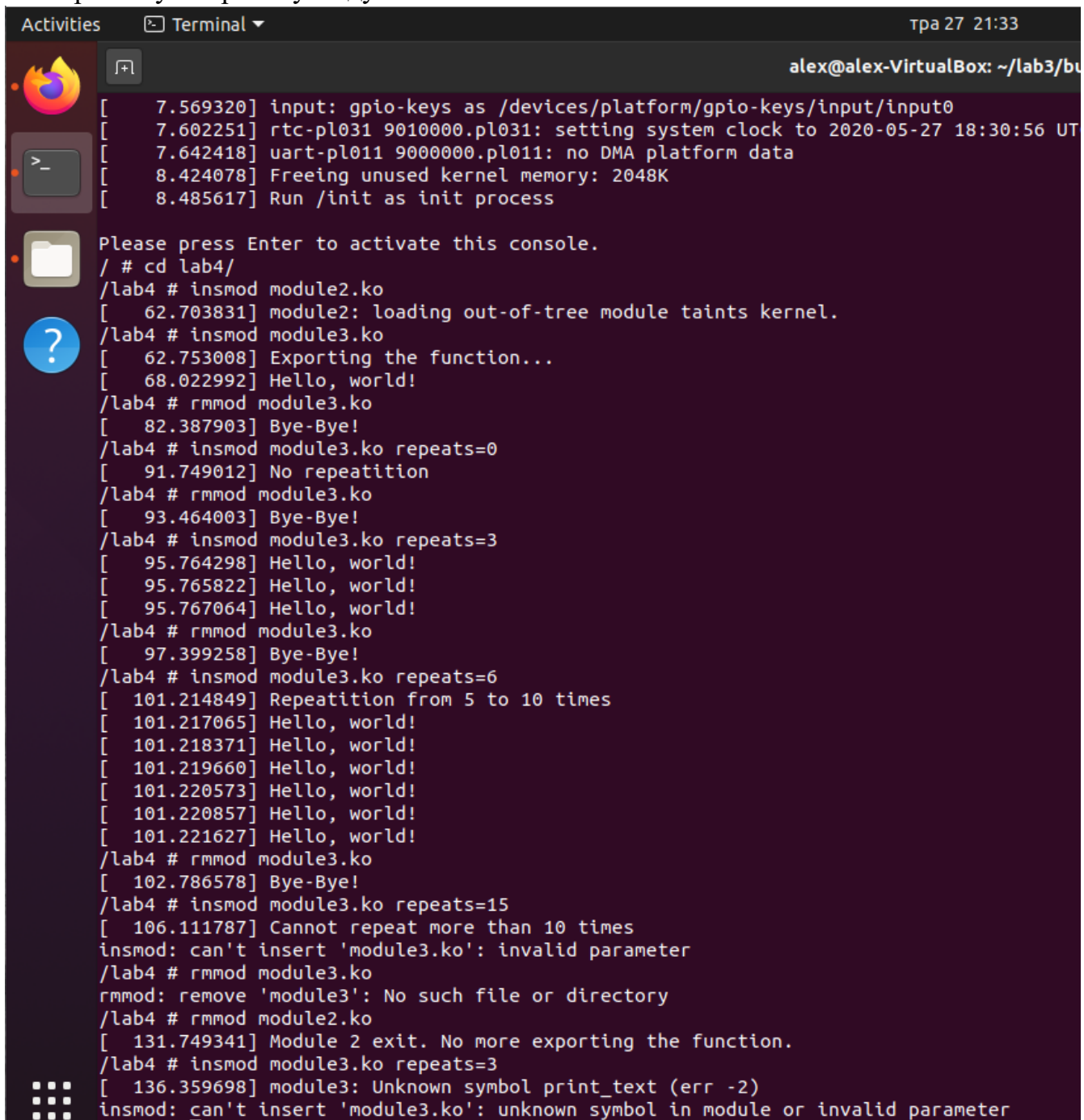
MODULE_AUTHOR("AlexHoncharenko");
MODULE_DESCRIPTION("Training to import a function");
MODULE_LICENSE("Dual BSD/GPL");
~
```

Виконуємо команду make для збирання модулю та створимо архів CPIO для rootfs та заархівуємо його за допомогою GZip:

```
Activities Terminal tpa 27 21:30 alex@alex-VirtualBox: ~/lab3/busybox

alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ export PATH=/opt/gcc-arm-8.3-2019.03-x86_64-arm-eabi/bin:$PATH
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ export CROSS_COMPILE='ccache arm-eabi-'
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ export ARCH=arm
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ export KDIR=/home/alex/lab3/linux-stable/
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ make
make -C /home/alex/lab3/linux-stable/ M=$PWD
make[1]: Entering directory '/home/alex/lab3/linux-stable'
CC [M] /home/alex/lab3/busybox/_install/lab4/module2.o
CC [M] /home/alex/lab3/busybox/_install/lab4/module3.o
Building modules, stage 2.
MODPOST 2 modules
CC /home/alex/lab3/busybox/_install/lab4/module2.mod.o
LD [M] /home/alex/lab3/busybox/_install/lab4/module2.ko
CC /home/alex/lab3/busybox/_install/lab4/module3.mod.o
LD [M] /home/alex/lab3/busybox/_install/lab4/module3.ko
make[1]: Leaving directory '/home/alex/lab3/linux-stable'
alex@alex-VirtualBox:~/lab3/busybox/_install/lab4$ cd ..
alex@alex-VirtualBox:~/lab3/busybox/_install$ find . | cpio -o -H newc | gzip > ../rootfs.cpio.gz
89868 blocks
alex@alex-VirtualBox:~/lab3/busybox/_install$ cd ..
alex@alex-VirtualBox:~/lab3/busybox$ qemu-system-arm -kernel _install/boot/zImage -initrd rootfs.cpio.gz -machine virt -nographic -m 512 --append "root=/dev/ram0 rw console=ttyAMA0,115200 mem=512M"
```


1. Протестуємо роботу модулів:



The screenshot shows a terminal window with a dark background. On the left, there are icons for Firefox, a terminal, and a help/question mark. The terminal title bar says 'alex@alex-VirtualBox: ~/lab3/bu'. The output shows the following sequence of events:

```
[ 7.569320] input: gpio-keys as /devices/platform/gpio-keys/input/input0
[ 7.602251] rtc-pl031 9010000.pl031: setting system clock to 2020-05-27 18:30:56 UT
[ 7.642418] uart-pl011 90000000.pl011: no DMA platform data
[ 8.424078] Freeing unused kernel memory: 2048K
[ 8.485617] Run /init as init process

Please press Enter to activate this console.
/ # cd lab4/
/lab4 # insmod module2.ko
[ 62.703831] module2: loading out-of-tree module taints kernel.
/lab4 # insmod module3.ko
[ 62.753008] Exporting the function...
[ 68.022992] Hello, world!
/lab4 # rmmod module3.ko
[ 82.387903] Bye-Bye!
/lab4 # insmod module3.ko repeats=0
[ 91.749012] No repeatition
/lab4 # rmmod module3.ko
[ 93.464003] Bye-Bye!
/lab4 # insmod module3.ko repeats=3
[ 95.764298] Hello, world!
[ 95.765822] Hello, world!
[ 95.767064] Hello, world!
/lab4 # rmmod module3.ko
[ 97.399258] Bye-Bye!
/lab4 # insmod module3.ko repeats=6
[ 101.214849] Repeatition from 5 to 10 times
[ 101.217065] Hello, world!
[ 101.218371] Hello, world!
[ 101.219660] Hello, world!
[ 101.220573] Hello, world!
[ 101.220857] Hello, world!
[ 101.221627] Hello, world!
/lab4 # rmmod module3.ko
[ 102.786578] Bye-Bye!
/lab4 # insmod module3.ko repeats=15
[ 106.111787] Cannot repeat more than 10 times
insmod: can't insert 'module3.ko': invalid parameter
/lab4 # rmmod module3.ko
rmmod: remove 'module3': No such file or directory
/lab4 # rmmod module2.ko
[ 131.749341] Module 2 exit. No more exporting the function.
/lab4 # insmod module3.ko repeats=3
[ 136.359698] module3: Unknown symbol print_text (err -2)
insmod: can't insert 'module3.ko': unknown symbol in module or invalid parameter
```

Помилка у виконанні запиту виникла, бо модулю `module3` не вдалося визначити символ `print_text`, який не був експортований в результаті незавантаження модулю `module2`, в якому цей символ реалізований та експортований.