

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота № 6
«Розроблення модулів Linux Kernel (частина 3)»

Виконав:
студент групи ІО-21
Пресняков А.В.
Перевірив:
Каплунов А.В.

Київ 2024

Завдання

Завдання розраховане на вже виконане завдання #3 (Лабораторна робота 4), і полягає у модифікації реалізації того завдання.

Зауваження:

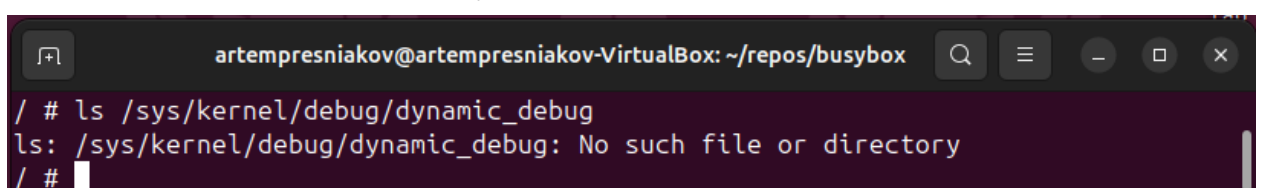
- I. Подивіться адреси завантаження своїх модулів на sysfs у каталозі поруч з параметрами модуля: /sys/module//sections/ А. Файли з адресами мають такі ж назви, як було названо секції, тобто починаються з крапки (.text, .init.text, ...), врахуйте це.
- II. Будь-ласка, додатково ознайомтесь з debugfs.
- III. Докладніше у додатку Debugging Techniques.
- IV. Для довідки:
 - A. \$KDIR/Documentation/admin-guide/dynamic-debug-howto.rst
 - B. \$KDIR/Documentation/filesystems/debugfs.txt
 - C. \$KDIR/Documentation/filesystems/proc.txt
 - D. \$KDIR/Documentation/filesystems/sysfs.txt
 - E. \$KDIR/Documentation/admin-guide/sysfs-rules.rst

Завдання 2

- I. Упевніться у відсутності каталогу: /sys/kernel/debug/dynamic_debug
 1. Це означає вимкнену опцію CONFIG_DYNAMIC_DEBUG (якщо збиралося по методичці, то не повинно бути).
- II. Замініть у функції exit модуля hello (hello1) друк вмісту списку на pr_debug і додайте два виклики pr_debug до та після друку списку.
- III. Перевірте залежність друку повідомлень від #define DEBUG на початку файлу.
- IV. Перезберіть ядро з увімкненим CONFIG_DYNAMIC_DEBUG, замініть його на nfs.
 1. Перезберіть модуль.
- V. Аналогічно показаному в appendix2, поекспериментуйте з друком з прапорцями p, f, m, а також зі встановленням їх для всього модуля та для окремих рядків.

Скріншоти виконання завдання

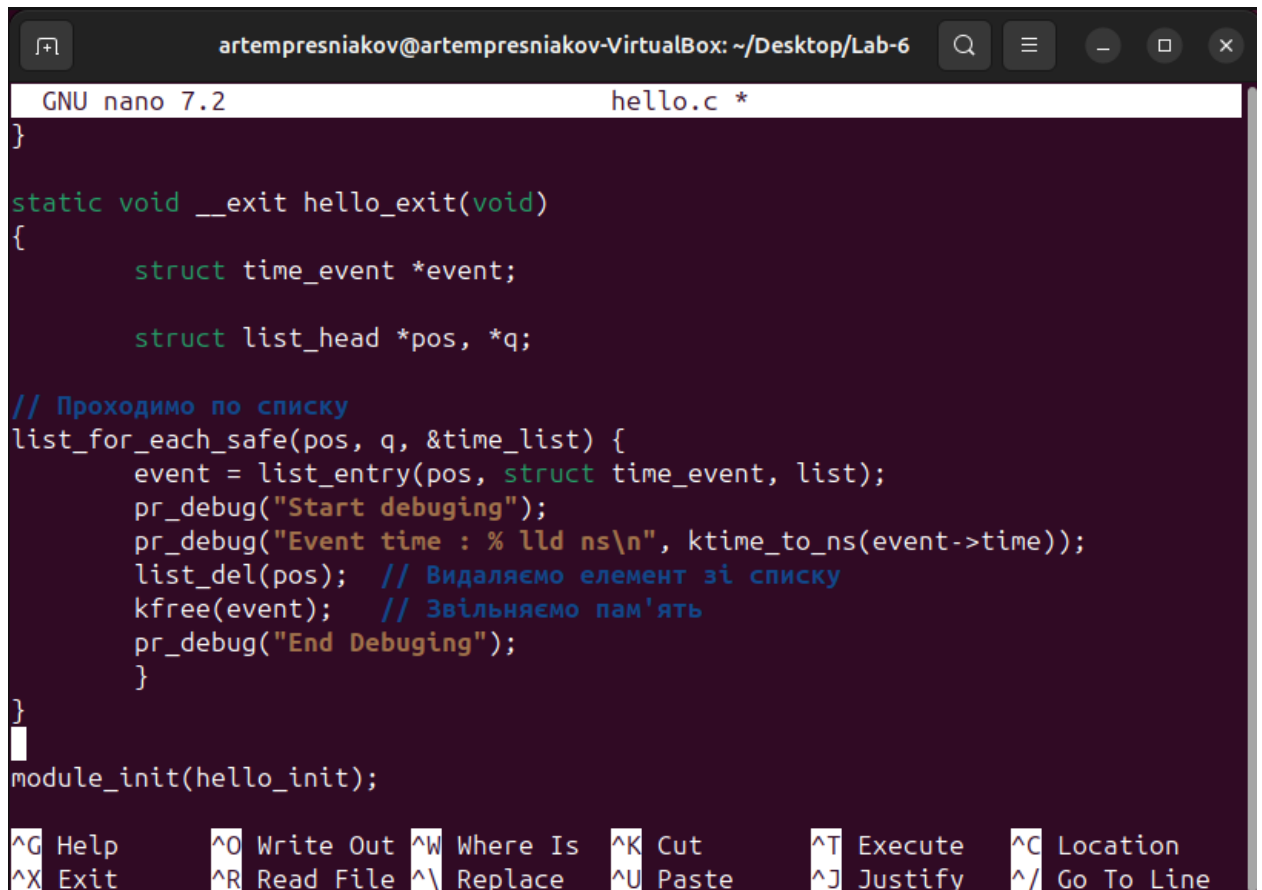
Впевнимось, що папки не існує:



```
artempresniakov@artempresniakov-VirtualBox: ~/repos/busybox
/ # ls /sys/kernel/debug/dynamic_debug
ls: /sys/kernel/debug/dynamic_debug: No such file or directory
/ #
```

Рисунок 1. Перевірка наявності папки dynamic_debug

Для виводу повідомлень до дебагу були замінені виклики `pr_info` на `pr_debug`.



```
artempresniakov@artempresniakov-VirtualBox: ~/Desktop/Lab-6
GNU nano 7.2 hello.c *
}

static void __exit hello_exit(void)
{
    struct time_event *event;

    struct list_head *pos, *q;

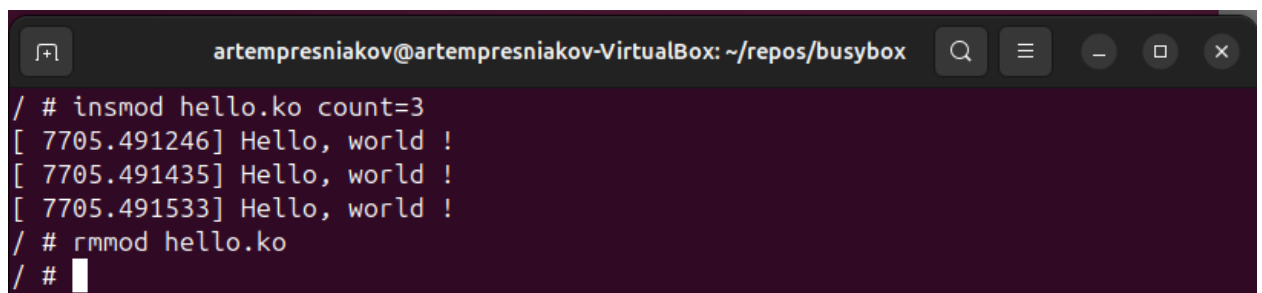
    // Проходимо по списку
    list_for_each_safe(pos, q, &time_list) {
        event = list_entry(pos, struct time_event, list);
        pr_debug("Start debugging");
        pr_debug("Event time : %lld ns\n", ktime_to_ns(event->time));
        list_del(pos); // Видаляємо елемент зі списку
        kfree(event); // Звільняємо пам'ять
        pr_debug("End Debuging");
    }
}

module_init(hello_init);

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Рисунок 2. Текст програми зі змінами

Якщо `#define DEBUG` відсутній у програмі, то відповідні повідомлення не виводяться (після виклику `rmmod`):

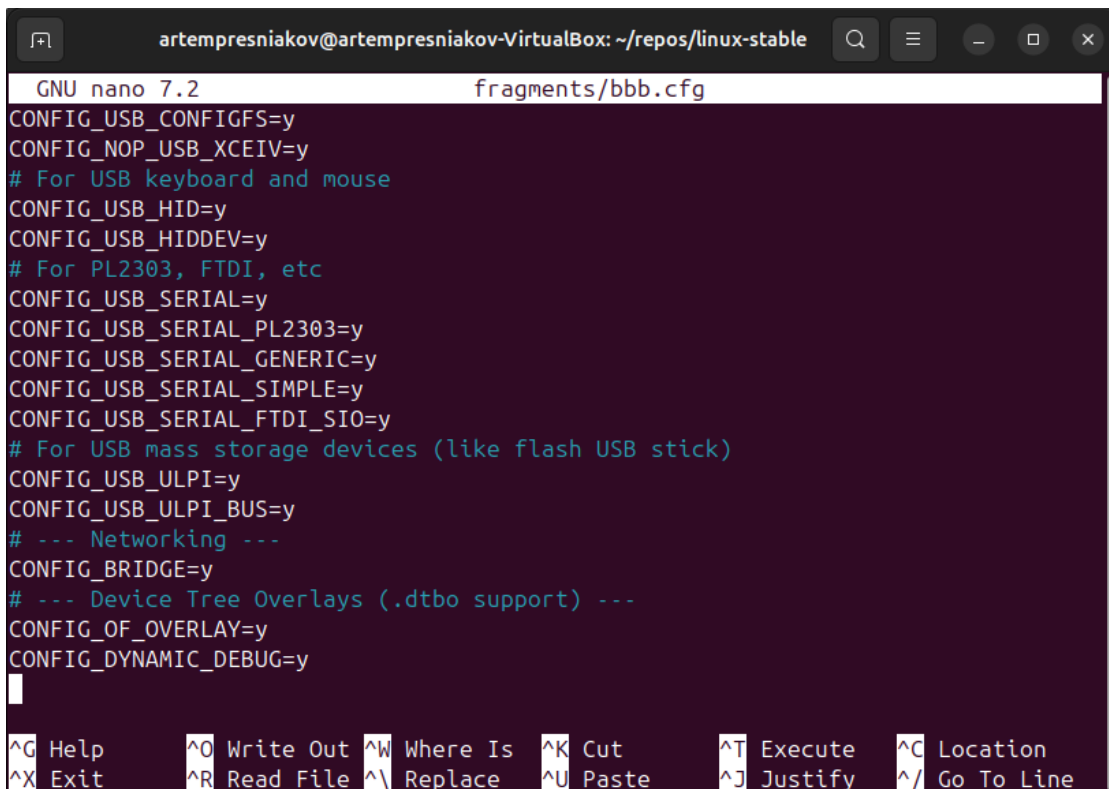


```
artempresniakov@artempresniakov-VirtualBox: ~/repos/busybox
/ # insmod hello.ko count=3
[ 7705.491246] Hello, world !
[ 7705.491435] Hello, world !
[ 7705.491533] Hello, world !
/ # rmmod hello.ko
/ #
```

Рисунок 3. Спроба виводу повідомлень

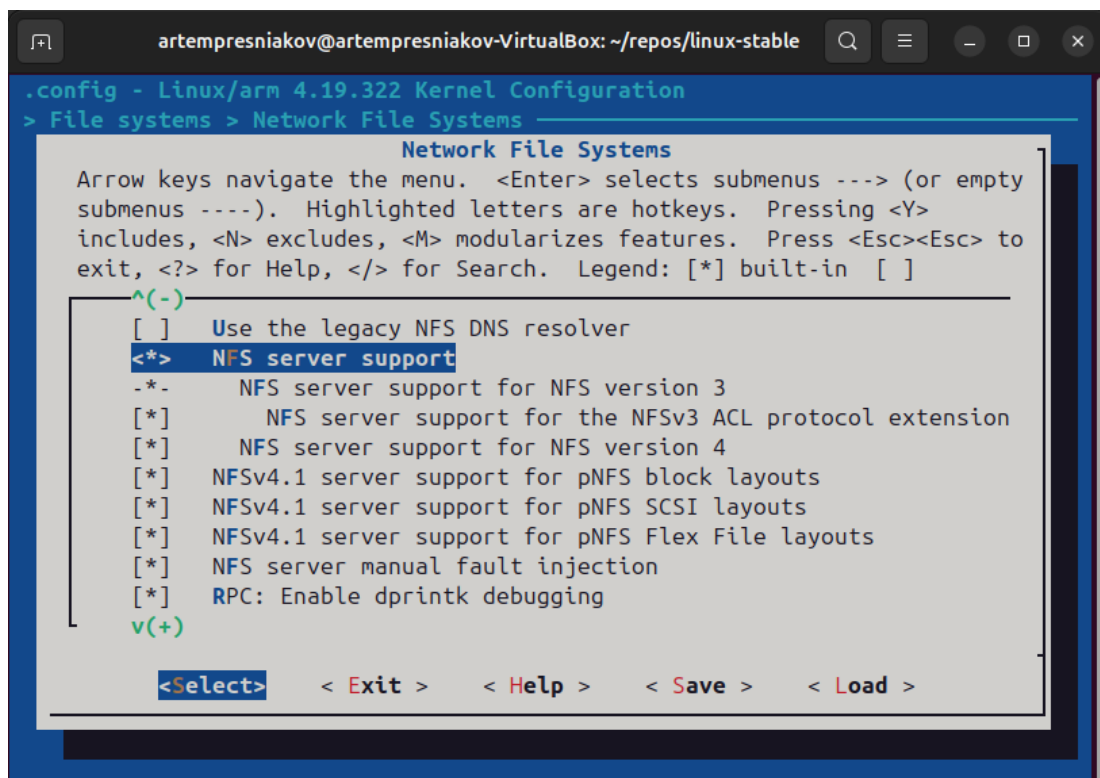
Якщо ж повторити відповідні дії з ввімкненим `debug`-режимом, то побачимо відповідні `debug` повідомлення у консолі.

Для того що б увімкнути `DEFINE_DYNAMIC_DEBUG` треба зробити відповідні зміни у ядрі системи та увімкнути `nfs` у налаштуваннях. Об'єднати параметри та скомпілювати ядро.



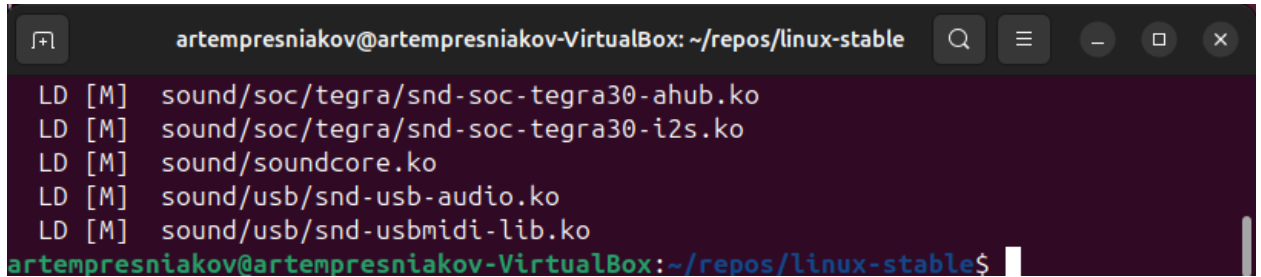
```
artempresniakov@artempresniakov-VirtualBox: ~/repos/linux-stable
GNU nano 7.2 fragments/bbb.cfg
CONFIG_USB_CONFIGFS=y
CONFIG_NOP_USB_XCEIV=y
# For USB keyboard and mouse
CONFIG_USB_HID=y
CONFIG_USB_HIDDEV=y
# For PL2303, FTDI, etc
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_PL2303=y
CONFIG_USB_SERIAL_GENERIC=y
CONFIG_USB_SERIAL_SIMPLE=y
CONFIG_USB_SERIAL_FTDI_SIO=y
# For USB mass storage devices (like flash USB stick)
CONFIG_USB_ULPI=y
CONFIG_USB_ULPI_BUS=y
# --- Networking ---
CONFIG_BRIDGE=y
# --- Device Tree Overlays (.dtbo support) ---
CONFIG_OF_OVERLAY=y
CONFIG_DYNAMIC_DEBUG=y
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рисунок 4. Увімкнення налаштувань ядра



```
.config - Linux/arm 4.19.322 Kernel Configuration
> File systems > Network File Systems
Network File Systems
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
^(-)
[ ] Use the legacy NFS DNS resolver
<*> NFS server support
.-. NFS server support for NFS version 3
[*] NFS server support for the NFSv3 ACL protocol extension
[*] NFS server support for NFS version 4
[*] NFSv4.1 server support for pNFS block layouts
[*] NFSv4.1 server support for pNFS SCSI layouts
[*] NFSv4.1 server support for pNFS Flex File layouts
[*] NFS server manual fault injection
[*] RPC: Enable dprintk debugging
v(+)
```

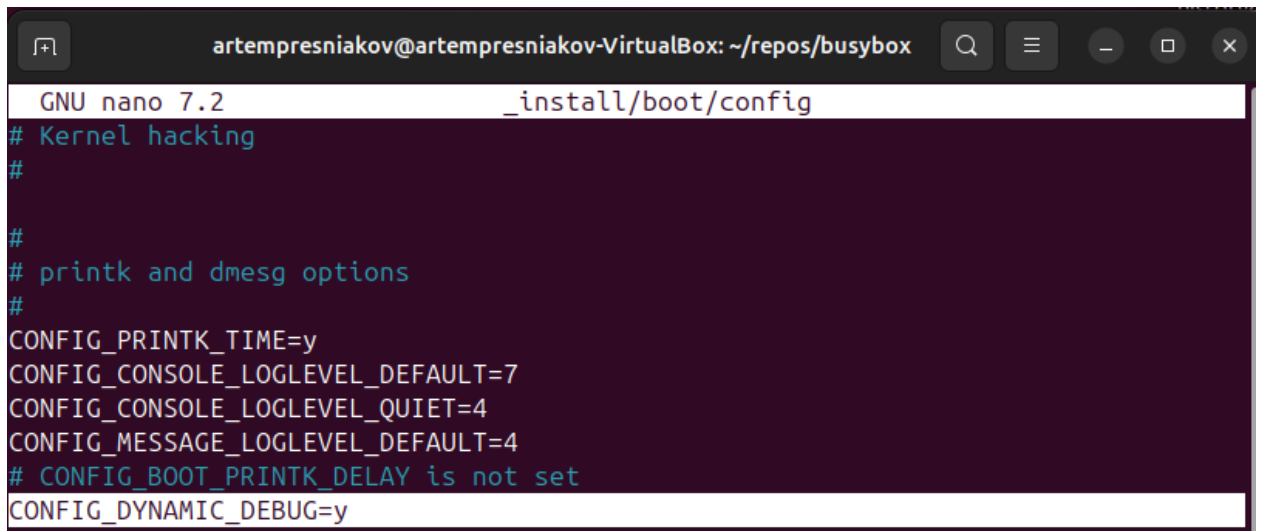
Рисунок 5. Налаштування nfs



```
artempresniakov@artempresniakov-VirtualBox: ~/repos/linux-stable
LD [M] sound/soc/tegra/snd-soc-tegra30-ahub.ko
LD [M] sound/soc/tegra/snd-soc-tegra30-i2s.ko
LD [M] sound/soundcore.ko
LD [M] sound/usb/snd-usb-audio.ko
LD [M] sound/usb/snd-usbmidi-lib.ko
artempresniakov@artempresniakov-VirtualBox:~/repos/linux-stable$
```

Рисунок 6. Збірка ядра

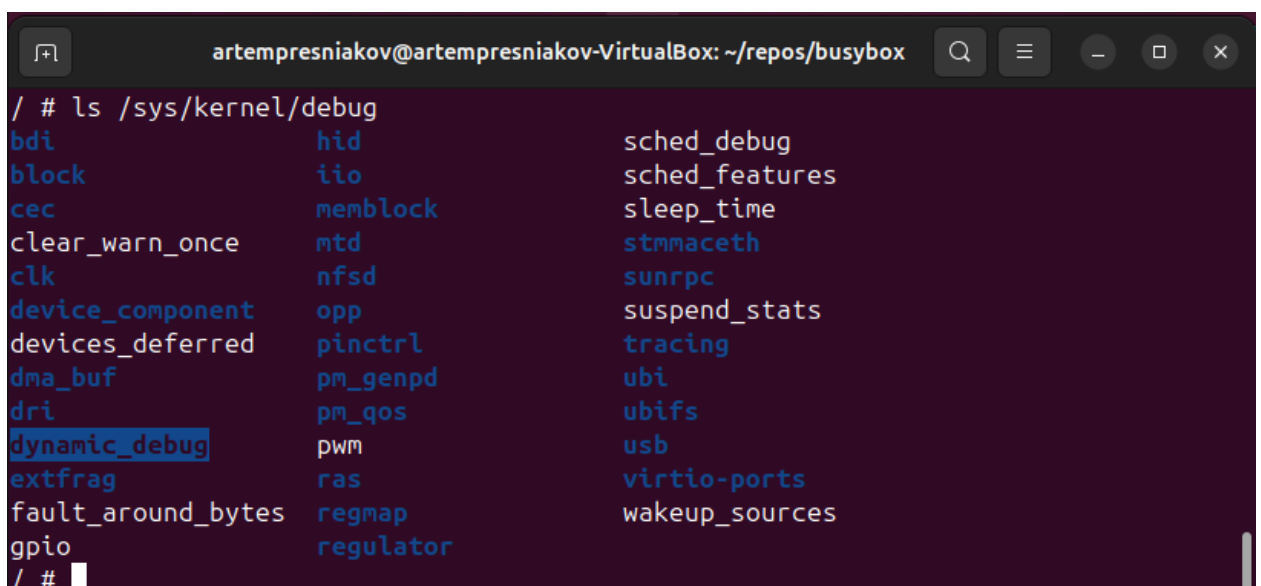
Також потрібно сконфігурувати сам busybox та перезібрати його.



```
artempresniakov@artempresniakov-VirtualBox: ~/repos/busybox
GNU nano 7.2 _install/boot/config
# Kernel hacking
#
#
# printk and dmesg options
#
CONFIG_PRINTK_TIME=y
CONFIG_CONSOLE_LOGLEVEL_DEFAULT=7
CONFIG_CONSOLE_LOGLEVEL_QUIET=4
CONFIG_MESSAGE_LOGLEVEL_DEFAULT=4
# CONFIG_BOOT_PRINTK_DELAY is not set
CONFIG_DYNAMIC_DEBUG=y
```

Рисунок 7. Увімкнення відповідного параметру у busybox

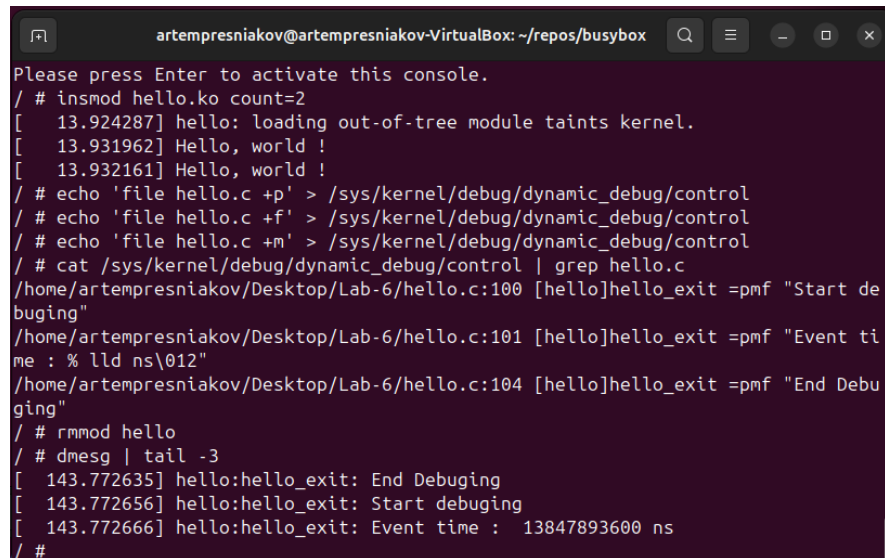
Після запуску цільової системи можемо впевнитись що відповідна папка з'явилась у каталогах.



```
artempresniakov@artempresniakov-VirtualBox: ~/repos/busybox
/ # ls /sys/kernel/debug
bdi          hid          sched_debug
block        iio          sched_features
cec          memblock    sleep_time
clear_warn_once mtd         stmmaceth
clk          nfsd         sunrpc
device_component opp         suspend_stats
devices_deferred pinctrl     tracing
dma_buf      pm_genpd    ubi
dri          pm_qos      ubifs
dynamic_debug pwm          usb
extfrag      ras         virtio-ports
fault_around_bytes regmap      wakeup_sources
gpio         regulator
/ #
```

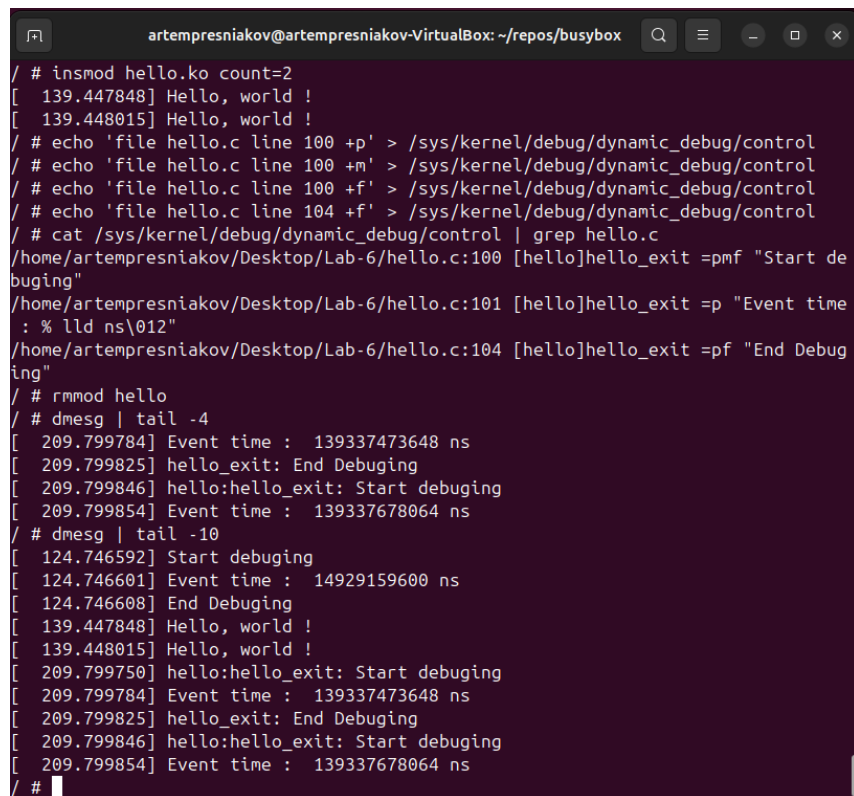
Рисунок 8. Перевірка наявності папки у кореневому каталозі

Тестування функціоналу полягало у встановленні на весь файл прапорців p, f, m. Флаг f відповідає за вивід ім'я функції, флаг m відповідає за вивід ім'я модуля, а p – за вивід відладкової інформації. Як можна побачити на фото 9 такі параметри були встановлені для всього файлу, і відповідно виведені повідомлення з цією інформацією.



```
artempresniakov@artempresniakov-VirtualBox: ~/repos/busybox
Please press Enter to activate this console.
/ # insmod hello.ko count=2
[ 13.924287] hello: loading out-of-tree module taints kernel.
[ 13.931962] Hello, world !
[ 13.932161] Hello, world !
/ # echo 'file hello.c +p' > /sys/kernel/debug/dynamic_debug/control
/ # echo 'file hello.c +f' > /sys/kernel/debug/dynamic_debug/control
/ # echo 'file hello.c +m' > /sys/kernel/debug/dynamic_debug/control
/ # cat /sys/kernel/debug/dynamic_debug/control | grep hello.c
/home/artempresniakov/Desktop/Lab-6/hello.c:100 [hello]hello_exit =pmf "Start de
buging"
/home/artempresniakov/Desktop/Lab-6/hello.c:101 [hello]hello_exit =pmf "Event ti
me : %lld ns\012"
/home/artempresniakov/Desktop/Lab-6/hello.c:104 [hello]hello_exit =pmf "End Debu
ging"
/ # rmmod hello
/ # dmesg | tail -3
[ 143.772635] hello:hello_exit: End Debuging
[ 143.772656] hello:hello_exit: Start debugging
[ 143.772666] hello:hello_exit: Event time : 13847893600 ns
/ #
```

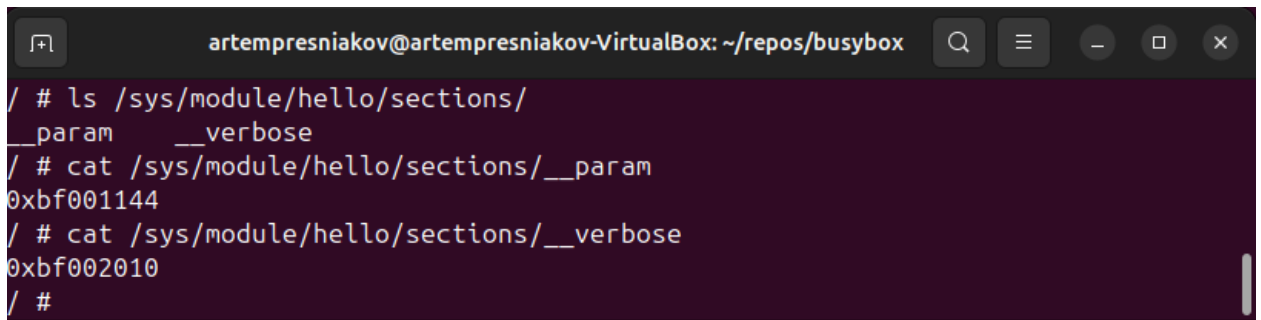
Рисунок 9. Тестування відладки, при застосуванні прапорців на весь файл



```
artempresniakov@artempresniakov-VirtualBox: ~/repos/busybox
/ # insmod hello.ko count=2
[ 139.447848] Hello, world !
[ 139.448015] Hello, world !
/ # echo 'file hello.c line 100 +p' > /sys/kernel/debug/dynamic_debug/control
/ # echo 'file hello.c line 100 +m' > /sys/kernel/debug/dynamic_debug/control
/ # echo 'file hello.c line 100 +f' > /sys/kernel/debug/dynamic_debug/control
/ # echo 'file hello.c line 104 +f' > /sys/kernel/debug/dynamic_debug/control
/ # cat /sys/kernel/debug/dynamic_debug/control | grep hello.c
/home/artempresniakov/Desktop/Lab-6/hello.c:100 [hello]hello_exit =pmf "Start de
buging"
/home/artempresniakov/Desktop/Lab-6/hello.c:101 [hello]hello_exit =p "Event time
: %lld ns\012"
/home/artempresniakov/Desktop/Lab-6/hello.c:104 [hello]hello_exit =pf "End Debug
ing"
/ # rmmod hello
/ # dmesg | tail -4
[ 209.799784] Event time : 139337473648 ns
[ 209.799825] hello_exit: End Debuging
[ 209.799846] hello:hello_exit: Start debugging
[ 209.799854] Event time : 139337678064 ns
/ # dmesg | tail -10
[ 124.746592] Start debugging
[ 124.746601] Event time : 14929159600 ns
[ 124.746608] End Debuging
[ 139.447848] Hello, world !
[ 139.448015] Hello, world !
[ 209.799750] hello:hello_exit: Start debugging
[ 209.799784] Event time : 139337473648 ns
[ 209.799825] hello_exit: End Debuging
[ 209.799846] hello:hello_exit: Start debugging
[ 209.799854] Event time : 139337678064 ns
/ #
```

Рисунок 10. Тестування відладки, при застосуванні прапорців на певні рядки

Як можна побачити на фото 10, для різних рядків були застосовані різні прапорці, що дозволяє отримати різні набори відладкової інформації.



```
artempresniakov@artempresniakov-VirtualBox: ~/repos/busybox
/ # ls /sys/module/hello/sections/
__param      __verbose
/ # cat /sys/module/hello/sections/__param
0xbf001144
/ # cat /sys/module/hello/sections/__verbose
0xbf002010
/ #
```

Рисунок 11. Адреси завантаження модулів на sysfs

Висновок

Виконавши лабораторну роботу №6 за завданням 2 було виконано увімкнення та використання динамічної відладки. Для цього, для початку, було перевірено наявність папки `dynamic_debug` – якщо її не має, як було у нашому випадку, то опція динамічної відладки вимкнена. Було модифіковано код, додано декілька нових виводів інформації через функції `pr_debug` та замінено `pr_info` на `pr_debug` у функції `hello_exit`. При перевірці залежності друку повідомлень від `#define DEBUG` на початку файлу було виявлено, що коли такий рядок присутній, то повідомлення від `pr_debug` можна побачити у консолі та журналі ядра, якщо ж такого рядка не має, то повідомлень ми не побачимо. Перезібравши ядро з увімкненим `CONFIG_DYNAMIC_DEBUG=y` та переконфігурувавши `busybox` ми ввімкнули дану опцію. Для перевірки цього твердження знову подивились на папку у корені операційної системи – вона там з'явилась. Також було перевірено поведінку програми з різними прапорцями та різними рівнями (для всього файлу та для певних рядків). З результатами можна ознайомитись на рисунках 9 та 10. На рисунку 11 представлено адреси завантажених модулів на `sysfs`. З огляду на результати, можна сказати що лабораторна робота виконана правильно.

ДОДАТОК А

https://github.com/6200211/CA-6-1_2118.git