|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**Лабораторная работа №9,10**

**по курсу “Операционные системы”**

**по теме “Обработчики прерываний”**

|  |  |
| --- | --- |
| Студент: | Уласик Е.А. |
| Группа: | ИУ7-61 |
| Преподаватель: | Рязанова Н.Ю. |

1. *г.*

**Задание 1 :**

* Написать загружаемый модуль ядра, в котором зарегистрировать

обработчик аппаратного прерывания с флагом IRQF\_SHARED.

* Инициализировать тасклет.
* В обработчике прерывания запланировать тасклет на выполнение.
* Вывести информацию о тасклете используя, или printk(), или seq\_file

interface

**Результат работы**:

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/init.h>

#include <linux/fs.h>

#include <linux/time.h>

#include <linux/slab.h>

#include <linux/interrupt.h>

#include <linux/time.h>

#define IRQ\_NUM 1

**char** my\_tasklet\_data[] = "TASKLET INFO";

**void** my\_tasklet\_function( **unsigned** **long** data);

**static** **int** irq\_counter = 0;

DECLARE\_TASKLET( my\_tasklet, my\_tasklet\_function, (**unsigned** **long**) &my\_tasklet\_data );

**void** my\_tasklet\_function( **unsigned** **long** data)

{

printk("%s:**\n** **\t** Tasklet state: %ld **\n** **\t** Tasklet count: %d **\n**",

(**char** \*)data, my\_tasklet.state, my\_tasklet.count);

**return**;

}

**static** **irqreturn\_t** irq\_handler(**int** irq, **void** \*dev, **struct** pt\_regs\* regs)

{

irq\_counter++;

**if** (irq == IRQ\_NUM)

{

tasklet\_schedule( &my\_tasklet );

**return** IRQ\_HANDLED;

}

**else**

**return** IRQ\_NONE;

}

Листинг 1. Программа задания 1 (Часть 1)

Листинг 2. программа задания 1 (Часть 2)

**static** **int** \_\_init tasklet\_init\_module(**void**)

{

printk(KERN\_INFO "Module was loaded");

**return** request\_irq(IRQ\_NUM, (**irq\_handler\_t**)irq\_handler, IRQF\_SHARED, "irq\_handler", (**void** \*)(irq\_handler));

}

**static** **void** \_\_exit tasklet\_exit\_module(**void**)

{

tasklet\_kill( &my\_tasklet );

free\_irq(IRQ\_NUM, (**void**\*)(irq\_handler));

printk(KERN\_INFO "Module was unloaded; **\n** **\t** Irq count: %d **\n**", irq\_counter);

}

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("Ulasik Evgeniy");

module\_init(tasklet\_init\_module);

module\_exit(tasklet\_exit\_module);

ifneq ($(KERNELRELEASE),)

obj-m := tasklet.o

else

CURRENT = **$(**shell uname -r**)**

KDIR = /lib/modules/**$(**CURRENT**)**/build

PWD = **$(**shell pwd**)**

default:

sudo **$(**MAKE**)** -C **$(**KDIR**)** M=**$(**PWD**)** modules

clean:

@rm -f \*.o .\*.cmd .\*.flags \*.mod.c \*.order

@rm -f .\*.\*.cmd \*~ \*.\*~ TODO.\*

@rm -fR .tmp\*

@rm -rf .tmp\_versions

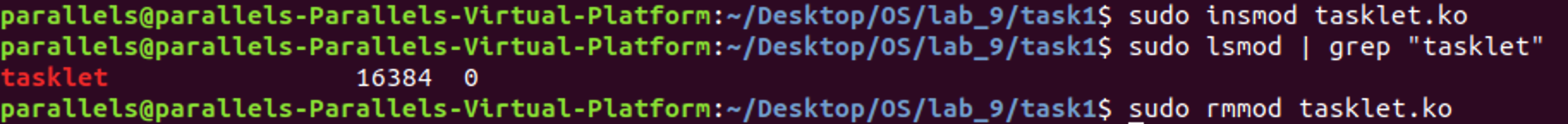
disclean: clean

@rm \*.ko \*.symvers

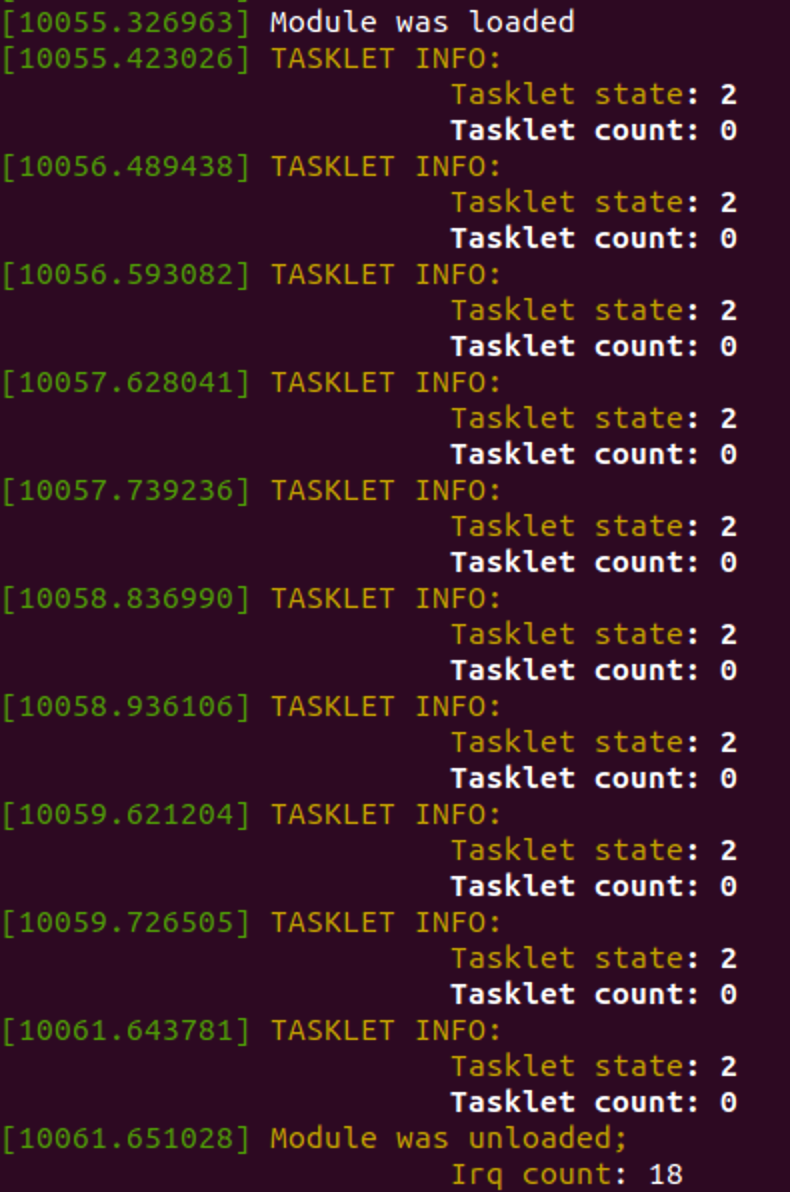
endif

Листинг 3. makefile

Подключение модуля, проверка подключения и выгрузка модуля:

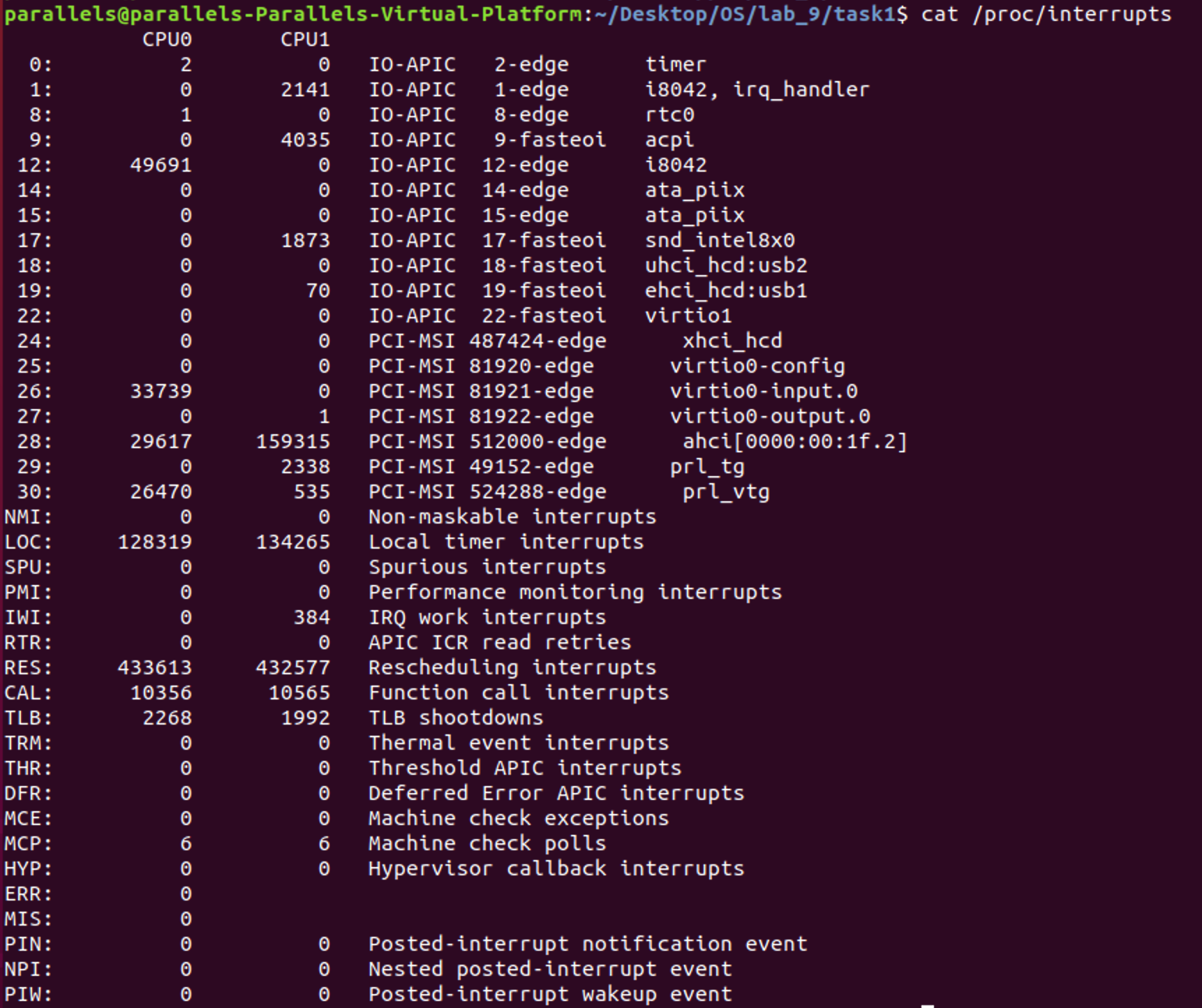


Вывод в сислог:



Если count = 0, то тасклет разрешён. Если он помечен как запланированный, то он может выполняться. Иначе тасклет запрещён и не может выполняться.

Разделение линии IRQ:



**Задание 2:**

* Написать загружаемый модуль ядра, в котором зарегистрировать

обработчик аппаратного прерывания с флагом IRQF\_SHARED.

* Инициализировать очередь работ.
* В обработчике прерывания запланировать очередь работ на

выполнение.

* Вывести информацию об очереди работ используя, или printk(), или

seq\_file interface

**Результат работы**:

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/init.h>

#include <linux/fs.h>

#include <linux/time.h>

#include <linux/slab.h>

#include <linux/interrupt.h>

#include <linux/time.h>

#include <linux/workqueue.h>

**static** **int** IRQ\_NUM = 1;

module\_param(IRQ\_NUM, **int**, S\_IRUGO);

**char** my\_tasklet\_data[] = "Work queue";

**struct** workqueue\_struct \*wq;

**void** hardwork\_function(**struct** work\_struct \*work);

**static** **int** irq\_counter = 0;

DECLARE\_WORK(hardwork, hardwork\_function);

**void** hardwork\_function(**struct** work\_struct \*work)

{

printk("work queue data: %d **\n**", work->data);

**return**;

}

**static** **irqreturn\_t** irq\_handler(**int** irq, **void** \*dev, **struct** pt\_regs\* regs)

{

irq\_counter++;

**if** (irq == IRQ\_NUM)

{

queue\_work(wq, &hardwork);

**return** IRQ\_HANDLED;

}

**else**

**return** IRQ\_NONE;

}

Листинг 4. Программа задания 2 (Часть 1)

Листинг 5. Программа задания 2 (Часть 2)

ifneq ($(KERNELRELEASE),)

obj-m := queue.o

else

CURRENT = **$(**shell uname -r**)**

KDIR = /lib/modules/**$(**CURRENT**)**/build

PWD = **$(**shell pwd**)**

default:

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

clean:

@rm -f \*.o .\*.cmd .\*.flags \*.mod.c \*.order

@rm -f .\*.\*.cmd \*~ \*.\*~ TODO.\*

@rm -fR .tmp\*

@rm -rf .tmp\_versions

disclean: clean

@rm \*.ko \*.symvers

endif

**static** **int** \_\_init wq\_init\_module(**void**)

{

**if** (request\_irq(IRQ\_NUM, (**irq\_handler\_t**)irq\_handler, IRQF\_SHARED, "work\_queue\_interrupt", (**void**\*)irq\_handler))

{

**return** -1;

}

printk(KERN\_INFO "Successfully loading ISR handler on IRQ %d**\n**", IRQ\_NUM);

wq = create\_workqueue("workqueue");

**if** (wq)

{

printk(KERN\_INFO "Work\_queue created!**\n**");

}

printk(KERN\_INFO "Module is now loaded!**\n**");**return** 0;

}

**static** **void** \_\_exit wq\_exit\_module(**void**)

{

flush\_workqueue(wq);

destroy\_workqueue(wq);

free\_irq(IRQ\_NUM, (**void**\*)irq\_handler);

printk("Successfully unloading, irq\_cnt = %d**\n**", irq\_counter);

printk("Module is now unloaded!**\n**");

**return**;

}

MODULE\_LICENSE("GPL");

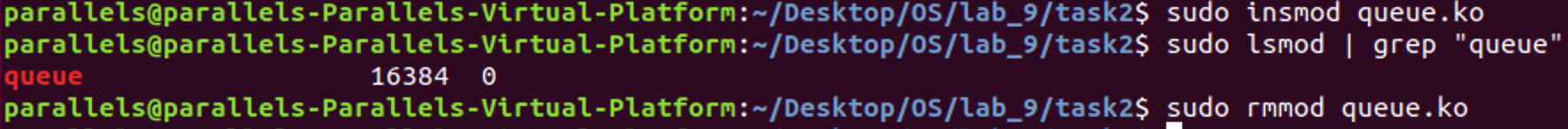
MODULE\_AUTHOR("Ulasik Evgeniy");

module\_init(wq\_init\_module);

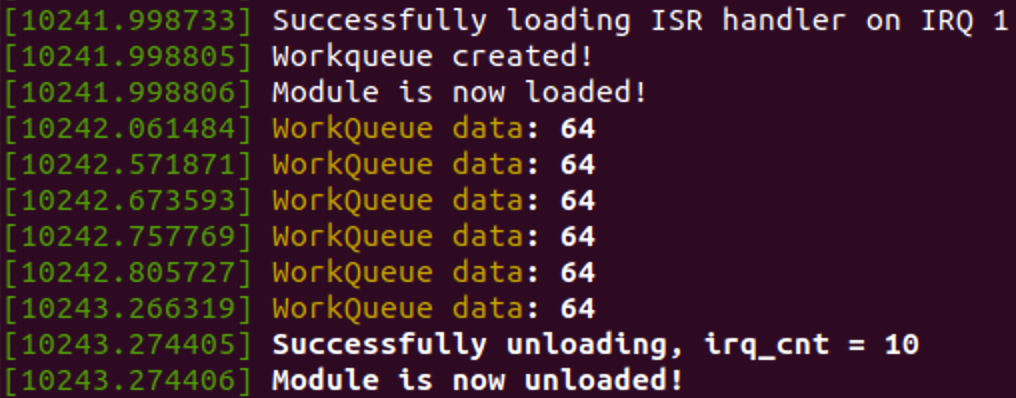
module\_exit(wq\_exit\_module);

Листинг 6. makefile

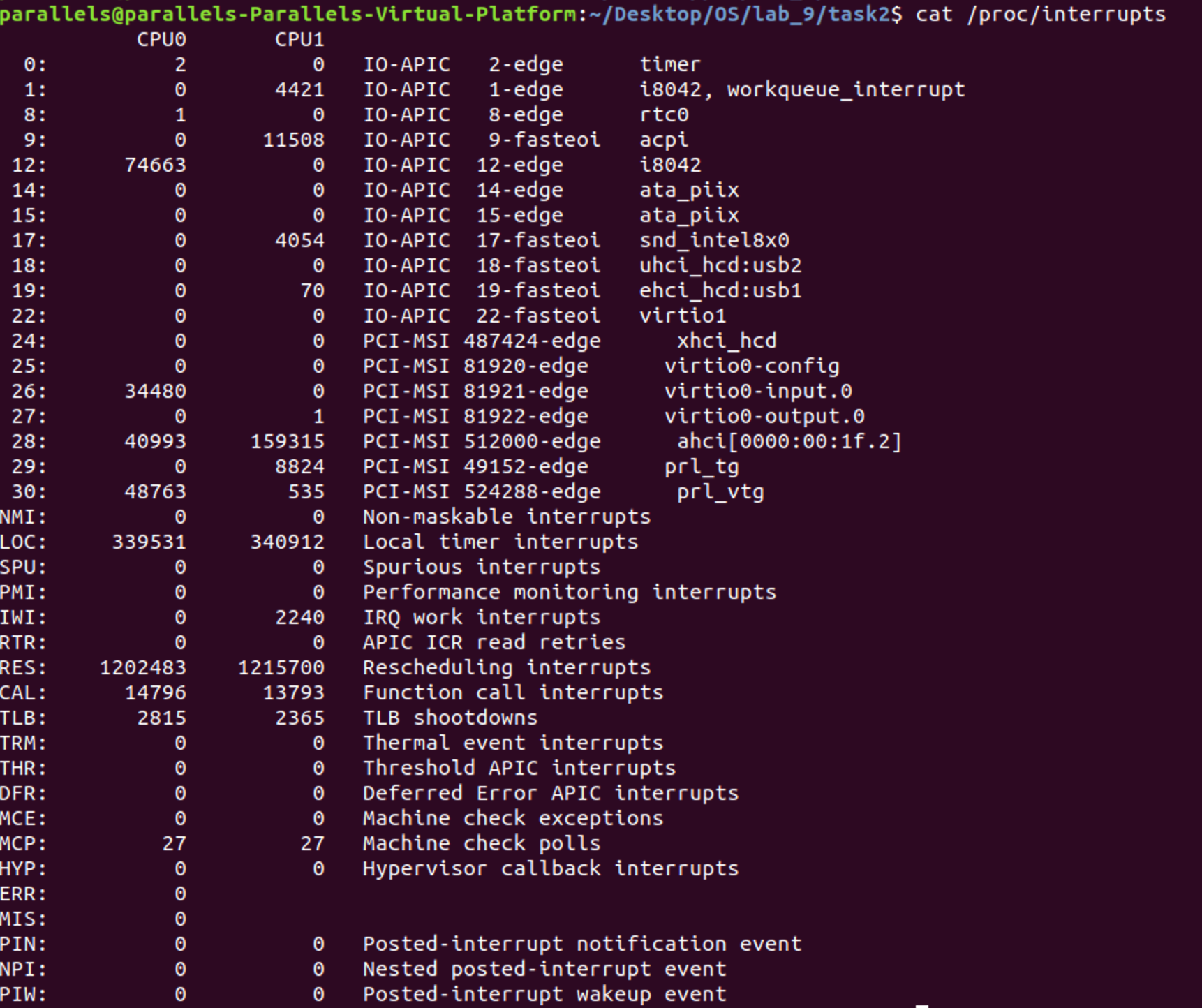
Подключение модуля, проверка подключения и выгрузка модуля:

****

Вывод в сислог:



Разделение линии IRQ:

****

Если загрузить оба модуля – tasklet.ko и queue.ko – в ядро, то можно наблюдать за разделением линии IRQ:

