|  |  |
| --- | --- |
|  | **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»

**ОТЧЕТ**

*к лабораторной работе №9*

*По курсу: «Операционные системы»*

*Тема: «Обработчики прерываний»*

Студент ИУ7-66Б

Турсунов Ж.Р.

Преподаватель

Рязанова Н.Ю.

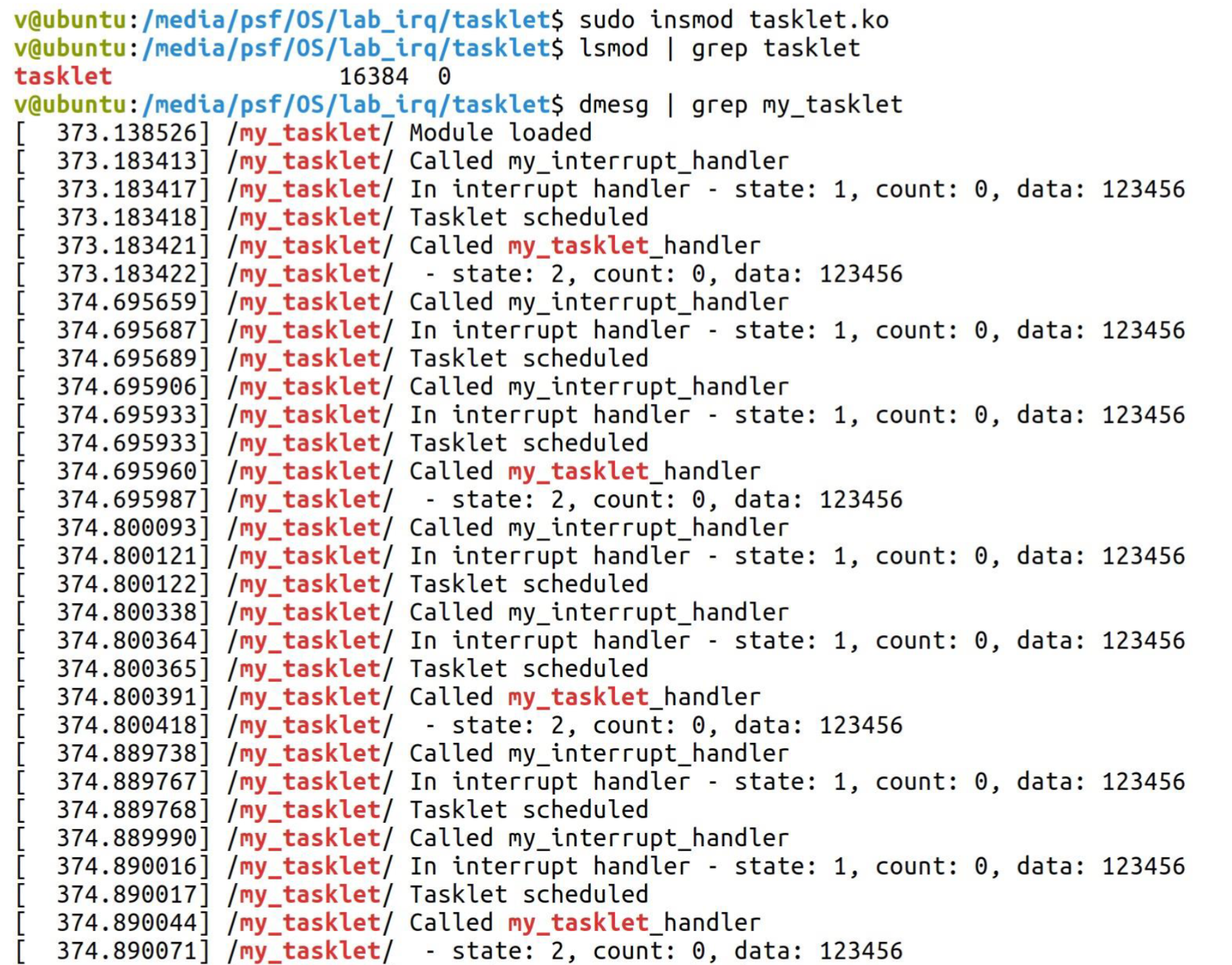
*Москва, 2021 г.*

**Часть 1: Тасклет**

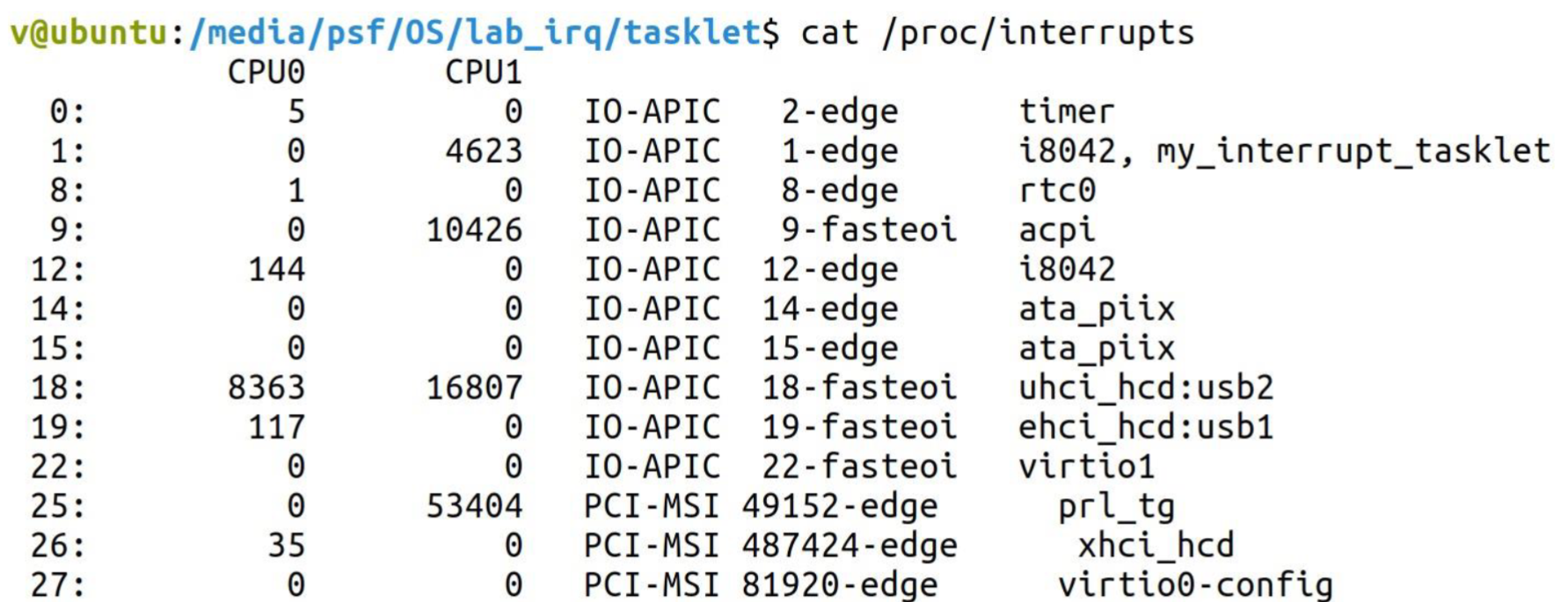
**Текст программы:**

|  |
| --- |
| #include <linux/kernel.h>  #include <linux/module.h>  #include <linux/interrupt.h>  #include <linux/seq\_file.h>  #include <linux/proc\_fs.h>  #define IRQ\_NUM 1  #define PROC\_FILE\_NAME "tasklet"  MODULE\_LICENSE**(**"GPL"**);**  MODULE\_AUTHOR**(**"Tursunov Jasurbek"**);**  static int my\_dev\_id**;**  const unsigned long my\_tasklet\_data **=** 123456**;**  static struct proc\_dir\_entry **\***proc\_file**;**  struct tasklet\_struct my\_tasklet**;**  static int \_\_init my\_tasklet\_init**(**void**);**  static int my\_tasklet\_open**(**struct inode **\***inode**,** struct file **\***file**);**  static int my\_tasklet\_release**(**struct inode **\***inode**,** struct file **\***file**);**  static int my\_tasklet\_show**(**struct seq\_file **\***m**,** void **\***v**);**  static void \_\_exit my\_tasklet\_exit**(**void**);**  void my\_tasklet\_handler**(**unsigned long data**);**  static irqreturn\_t my\_interrupt\_handler**(**int irq**,** void **\***dev\_id**);**  DECLARE\_TASKLET**(**my\_tasklet**,** my\_tasklet\_handler**,** my\_tasklet\_data**);**  static const struct proc\_ops my\_tasklet\_props **=**  **{**  **.**proc\_open **=** my\_tasklet\_open**,**  **.**proc\_release **=** my\_tasklet\_release**,**  **.**proc\_read **=** seq\_read  **};**  static inline void printk\_tasklet\_info**(**const char**\*** prefix**)**  **{**  printk**(**KERN\_INFO "/my\_tasklet/ %s -state: %ld, count: %d, data: %ld\n"**,**prefix**,** my\_tasklet**.**state**,** my\_tasklet**.**count**,** my\_tasklet**.**data**);**  **}**  static int my\_tasklet\_show**(**struct seq\_file **\***m**,** void **\***v**)**  **{**  printk**(**KERN\_INFO "/my\_tasklet/ Called my\_tasklet\_show\n"**);**  seq\_printf**(**m**,** "state: %ld\ncount: %d\ndata: %ld\n"**,** my\_tasklet**.**state**,** my\_tasklet**.**count**,** my\_tasklet**.**data**);**  **return** 0**;**  **}**  static int my\_tasklet\_open**(**struct inode **\***inode**,** struct file **\***file**)**  **{**  printk**(**KERN\_INFO "/my\_tasklet/ Called my\_tasklet\_open\n"**);**  **return** single\_open**(**file**,** my\_tasklet\_show**,** **NULL);**  **}**  void my\_tasklet\_handler**(**unsigned long data**)**  **{**  printk**(**KERN\_INFO "/my\_tasklet/ Called my\_tasklet\_handler\n"**);**  printk\_tasklet\_info**(**""**);**  **}**  static irqreturn\_t my\_interrupt\_handler**(**int irq**,** void **\***dev\_id**)**  **{**  printk**(**KERN\_INFO "/my\_tasklet/ Called my\_interrupt\_handler\n"**);**  **if** **(**irq **==** IRQ\_NUM**)**  **{**  tasklet\_schedule**(&**my\_tasklet**);**  printk\_tasklet\_info**(**"In interrupt handler"**);**  printk**(**KERN\_INFO "/my\_tasklet/ Tasklet scheduled\n"**);**  **return** IRQ\_HANDLED**;**  **}**  **else** **return** IRQ\_NONE**;**  **}**  static int my\_tasklet\_release**(**struct inode **\***inode**,** struct file **\***file**)**  **{**  printk**(**KERN\_INFO "/my\_tasklet/ Called my\_tasklet\_release\n"**);**  **return** single\_release**(**inode**,** file**);**  **}**  static int \_\_init my\_tasklet\_init**(**void**)**  **{**  proc\_file **=** proc\_create**(**PROC\_FILE\_NAME**,** S\_IRUGO**,** **NULL,** **&**my\_tasklet\_props**);**  **if** **(!**proc\_file**)**  **{**  printk**(**KERN\_ERR "/my\_tasklet/ Error: can't create seq file\n"**);**  **return** **-**ENOMEM**;**  **}**  **if** **(**request\_irq**(**IRQ\_NUM**,** my\_interrupt\_handler**,** IRQF\_SHARED**,** "my\_interrupt\_tasklet"**,** **&**my\_dev\_id**))**  **{**  printk**(**KERN\_ERR "/my\_tasklet/ Error: can't register irq handler\n"**);**  **return** **-**1**;**  **}**  printk**(**KERN\_INFO "/my\_tasklet/ Module loaded\n"**);**  **return** 0**;**  **}**  static void \_\_exit my\_tasklet\_exit**(**void**)**  **{**  tasklet\_kill**(&**my\_tasklet**);**  free\_irq**(**IRQ\_NUM**,** **&**my\_dev\_id**);**  **if** **(**proc\_file**)**  remove\_proc\_entry**(**PROC\_FILE\_NAME**,** **NULL);**  printk**(**KERN\_INFO "/my\_tasklet/ Module unloaded\n"**);**  **}**  module\_init**(**my\_tasklet\_init**);**  module\_exit**(**my\_tasklet\_exit**);** |

**Результат работы**



**Рис 1: Загрузка модуля**



**Рис 2: Содержимое файла /proc/interrupts:**



**Рис 3: Информация о тасклете в user через seqфайлы:**



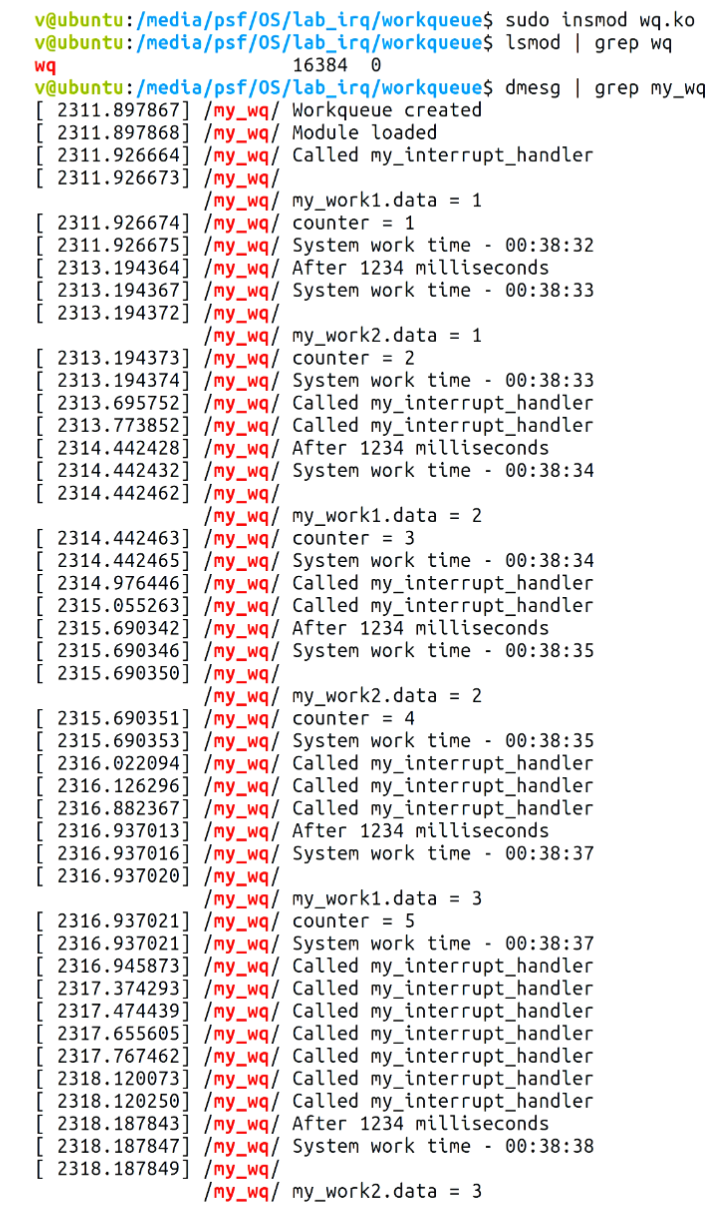
**Рис 4: Выгрузка модуля**

**Часть 2: Очередь работ**

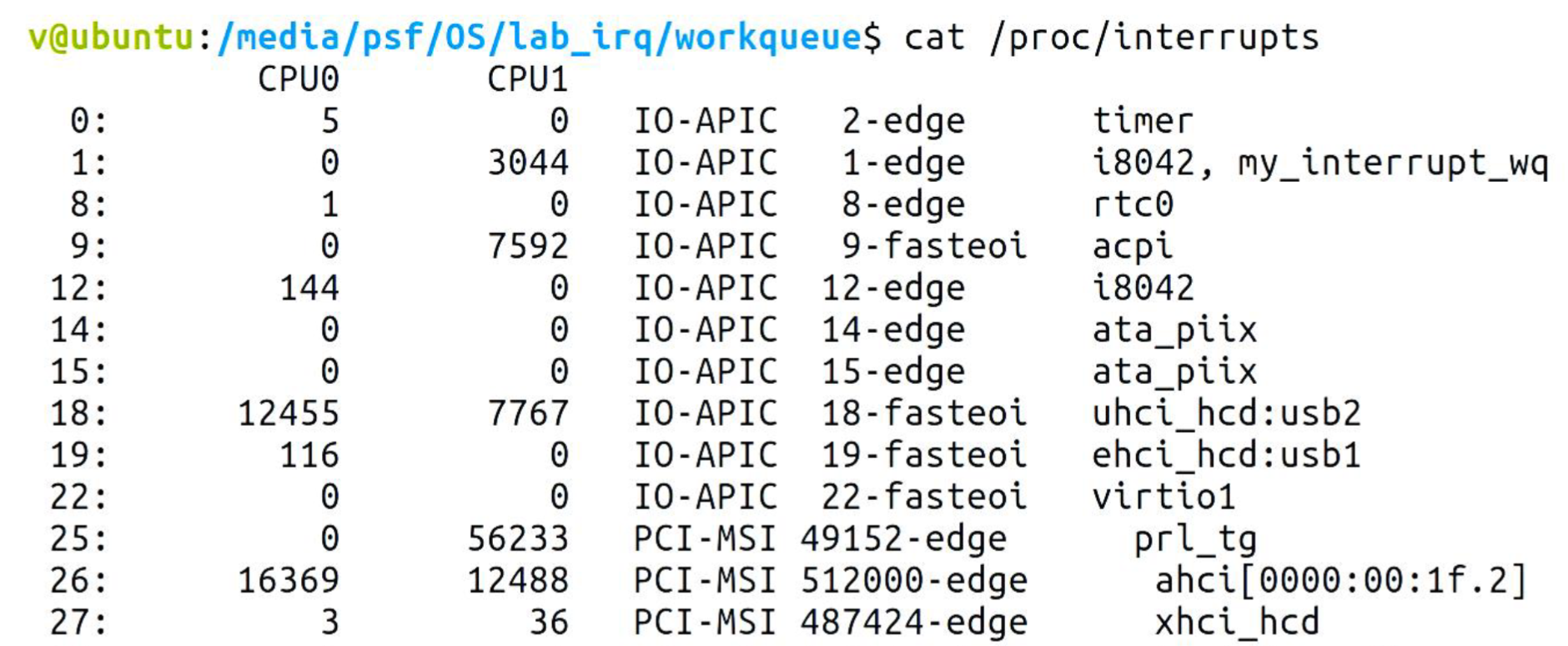
**Текст программы:**

|  |
| --- |
| #include <linux/module.h>  #include <linux/interrupt.h>  #include <linux/workqueue.h>  #include <linux/slab.h>  #define IRQ\_NUM 1  MODULE\_LICENSE**(**"GPL"**);**  MODULE\_AUTHOR**(**"Tursunov Jasurbek"**);**  static int my\_dev\_id**;**  static int irq\_call\_count**=** 0**;**  struct workqueue\_struct **\***wq**;**  **typedef** struct  **{**  struct work\_struct my\_work**;**  int id**;**  int data**;**  **}**  my\_work\_t**;**  my\_work\_t **\***work1**,** **\***work2**;**  void printk\_time**(**void**)**  **{**  struct timespec64 ts**;**  ktime\_get\_ts64**(&**ts**);**  printk**(**KERN\_INFO "/my\_wq/ System work time -%02lld:%02lld:%02lld\n"**,** ts**.**tv\_sec **/** 3600 **%** 24**,** ts**.**tv\_sec **/** 60 **%** 60**,** ts**.**tv\_sec **%** 60**);**  **}**    void my\_wq\_function**(**struct work\_struct **\***work**)**  **{**  int delay **=** 1234**;**  my\_work\_t **\***my\_work **=** **(**my\_work\_t **\*)** work**;**  printk**(**KERN\_INFO "/my\_wq/\n/my\_wq/ my\_work%d.data = %d\n"**,** my\_work**->**id**,** **++**my\_work**->**data**);**  printk**(**KERN\_INFO "/my\_wq/ counter = %d\n"**,** **++**irq\_call\_n**);**  printk\_time**();**  msleep**(**delay**);**  printk**(**KERN\_INFO "/my\_wq/ After %d milliseconds\n"**,** delay**);**  printk\_time**();**  **}**  static irqreturn\_t my\_interrupt\_handler**(**int irq**,** void **\***dev\_id**)**  **{**  **if** **(**irq **==** SHARED\_IRQ**)**  **{**  printk**(**KERN\_INFO "/my\_wq/ Called my\_interrupt\_handler\n"**);**  **if** **(**work1**)**  queue\_work**(**wq**,** **&**work1**->**my\_work**);**  **if** **(**work2**)**  queue\_work**(**wq**,** **&**work2**->**my\_work**);**  **return** IRQ\_HANDLED**;**  **}**  **else**  **return** IRQ\_NONE**;**  **}**  static my\_work\_t**\*** alloc\_work**(**int id**)**  **{**  my\_work\_t**\***work **=** **(**my\_work\_t **\*)** kmalloc**(sizeof(**my\_work\_t**),** GFP\_KERNEL**);**  **if** **(**work**)**  **{**  work**->**id **=** id**;**  work**->**data **=** 0**;**  **}**  **return** work**;**  **}**  static int \_\_init my\_wq\_init**(**void**)**  **{**  **if** **(**request\_irq**(**IRQ\_NUM**,** my\_interrupt\_handler**,** IRQF\_SHARED**,** "my\_interrupt\_wq"**,** **&**my\_dev\_id**))**  **{**  printk**(**KERN\_ERR "/my\_wq/Error: can't register interrupt handler\n"**);**  **return** **-**1**;**  **}**  wq**=** create\_workqueue**(**"workqueue"**);**  **if** **(**wq**)**  **{**  printk**(**KERN\_INFO "/my\_wq/ Workqueue created\n"**);**  **if** **((**work1 **=** alloc\_work**(**1**)))**  INIT\_WORK**(&**work1**->**my\_work**,** my\_wq\_function**);**  **else**  printk**(**KERN\_ERR "/my\_wq/ Error: can't initwork1\n"**);**  **if** **((**work2 **=** alloc\_work**(**2**)))**  INIT\_WORK**(&**work2**->**my\_work**,** my\_wq\_function**);**  **else**  printk**(**KERN\_ERR "/my\_wq/ Error: can't init work2\n"**);**  **}**  **else**  **{**  free\_irq**(**IRQ\_NUM**,** **&**my\_dev\_id**);**  printk**(**KERN\_ERR "/my\_wq/ Error: can't create workqueue\n"**);**  **return** **-**ENOMEM**;**  **}**  printk**(**KERN\_INFO "/my\_wq/ Module loaded\n"**);**  **return** 0**;**  **}**  static void \_\_exit my\_wq\_exit**(**void**)**  **{**  flush\_workqueue**(**wq**);**  destroy\_workqueue**(**wq**);**  free\_irq**(**IRQ\_NUM**,** **&**my\_dev\_id**);**  **if** **(**work1**)**  kfree**((**void **\*)**work1**);**  **if** **(**work2**)**  kfree**((**void **\*)**work2**);**  printk**(**KERN\_INFO "/my\_wq/ Module unloaded\n"**);**  **}**  module\_init**(**my\_wq\_init**);**  module\_exit**(**my\_wq\_exit**);** |

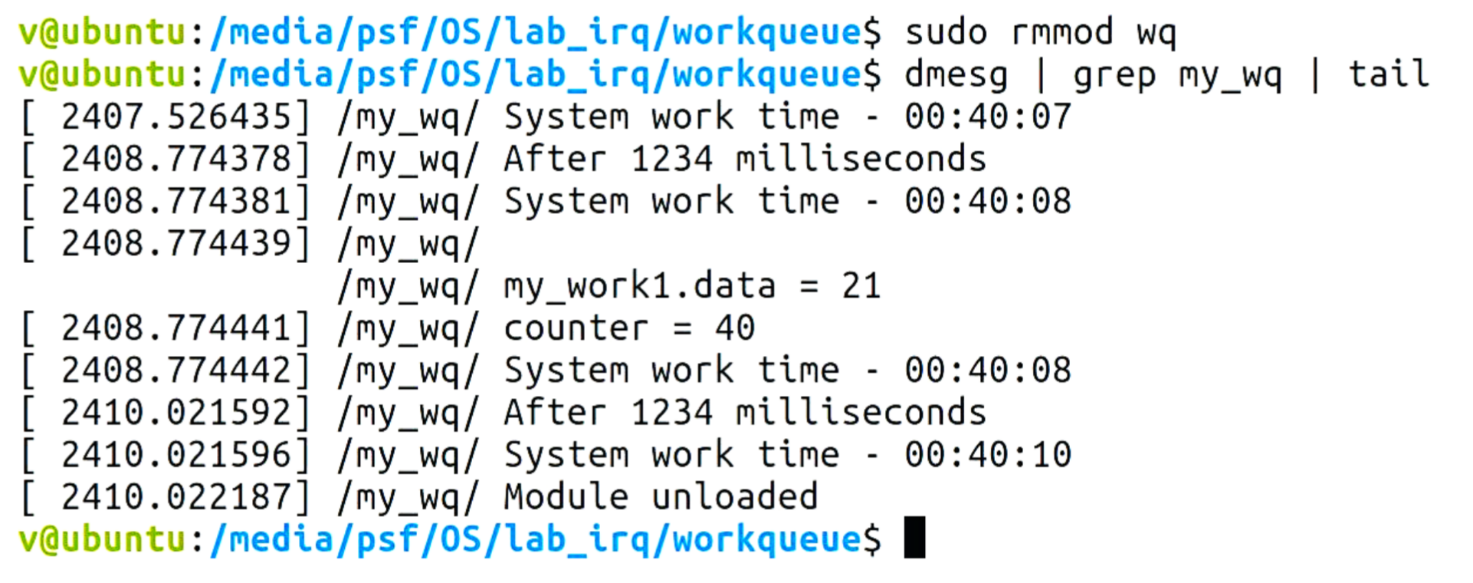
**Результат работы**



**Рис 1: Загрузка модуля**



**Рис 2: Содержимое файла /proc/interrupts:**



**Рис 3: Выгрузка модуля**