|  |  |
| --- | --- |
|  | **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

Дедич Б.Б

Преподаватель

Рязанова Н.Ю.

*Москва, 2021 г.*

Задание 1.

Код программы представлен на листинге 1.

|  |
| --- |
| Листинг1. tasklet.c  #include <linux/module.h>  #include <linux/kernel.h>  #include <linux/init.h>  #include <linux/interrupt.h>  #include <linux/timex.h>  MODULE\_LICENSE("GPL");  MODULE\_AUTHOR**(**"Dedich Borislav Borislavovich"**);**  static int my\_irq = 1, my\_dev\_id, irq\_cnt = 0;  void **tasklet\_function**(unsigned long data);  char my\_tasklet\_data[] = "my\_tasklet\_function was called";  DECLARE\_TASKLET(my\_tasklet, tasklet\_function, (unsigned long)&my\_tasklet\_data);  void **tasklet\_function**(unsigned long data) {  printk(KERN\_INFO "++ tasklet\_function was called. state '%ld' count '%i' data '%s'\n", my\_tasklet.state, my\_tasklet.count, (char\*)data);  }  static irqreturn\_t **my\_irq\_handler**(int irq, void \*dev) {  if (irq == my\_irq) {  printk(KERN\_INFO "++ my\_irq\_handler was called %d time(s)\n", ++irq\_cnt);  tasklet\_schedule(&my\_tasklet);  return IRQ\_HANDLED;  }  return IRQ\_NONE;  }  static int \_\_init **my\_tasklet\_init**(void) {  if (request\_irq(my\_irq, my\_irq\_handler, IRQF\_SHARED, "my\_tasklet", &my\_dev\_id)) {  printk(KERN\_ERR "++ can't get assigned IRQ %i\n", my\_irq);  return 1;  }  printk(KERN\_INFO "++ Successfully loaded handler for IRQ %d\n", my\_irq);  return 0;  }  static void \_\_exit **my\_tasklet\_exit**(void) {  tasklet\_kill(&my\_tasklet);  synchronize\_irq(my\_irq);  free\_irq(my\_irq, &my\_dev\_id);  printk(KERN\_INFO "++ tasklet unloaded, irq\_counter = %d\n", irq\_cnt);  return;  }  module\_init(my\_tasklet\_init);  module\_exit(my\_tasklet\_exit); |

Был написан обработчик прерываний для irq 1 (клавиатура).

Загрузим модуль и проверим журнал. Результаты представлены на рисунке 1.

|  |
| --- |
| Рис 1. Состояние системного журнала после загрузки модуля |

При каждом прерывании от клавиатуры вызывается обработчик прерывания my\_irq\_handler. В нем вызывается tasklet\_schedule(). При обработке тасклета выводится информация о нем.

Проверим, что my\_tasklet добавилось в список прерываний. На рисунке 2 представлено содержимое файла /proc/interrupts, что говорит о том, что my\_tasklet добавилось в список прерываний.

|  |
| --- |
| Рис 2. Содержимое файла /proc/interrupts |

Выгрузим модуль и проверим содержимое системного журнала. Результаты представлены на рисунке 3.

|  |
| --- |
| Рис 3. Состояние системного журнала после выгрузки модуля. |

Покажем, что модуль удален. На рисунке 4 представлено содержание файла /proc/interrupts, что говорит о том, что my\_tasklet удалилось из списка прерывания после выгрузки модуля.

|  |
| --- |
| Рис 4. Содержимое файла /proc/interrupts |

**Задание 2.**

Код программы представлен на листинге 2.

|  |
| --- |
| Листинг 2. queue.c  #include <linux/module.h>  #include <linux/interrupt.h>  #include <linux/workqueue.h>  #include <linux/slab.h>  #include <linuc/delay.h>  #define IRQ\_NUM 1  MODULE\_LICENSE**(**"GPL"**);**  MODULE\_AUTHOR**(**"Dedich Borislav Borislavovich"**);**  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**);** |

Загрузим модуль и проверим журнал. Результаты представлены на рисунке 5.

|  |
| --- |
| Рис 5. Состояние системного журнала после загрузки модуля |

Проверим, что my\_int\_worqueue добавилось в список прерываний. На рисунке 6 представлено содержимое файла /proc/interrupts, что говорит о том, что my\_interrupt\_wq добавилось в список прерываний.

|  |
| --- |
| Рис 6. Содержимое файла /proc/interrupts |

Выгрузим модуль и проверим содержимое системного журнала. Результаты представлены на рисунке 3.

|  |
| --- |
| Рис 7. Состояние системного журнала после выгрузки модуля. |

Покажем, что модуль удален. На рисунке 8 представлено содержание файла /proc/interrupts, что говорит о том, что my\_interrupt\_wq удалилось из списка прерывания после выгрузки модуля.

|  |
| --- |
| Рис 8. Содержимое файла /proc/interrupts |