МГТУ им. Баумана

Лабораторная работа №9 часть 2

По курсу: "Операционные системы"

«Обработчики прерываний. Очереди работ»

Работу выполнил: студент группы ИУ7-63Б
Наместник Анастасия

Преподаватель: Рязанова Н. Ю.

Москва, 2021

На листинге 1 представлена программа, реализующая создание одной очереди работ и добавление в нее 2-ух работ, выполняющих разные задачи. Обработчик прерываний irq_handler считывает скан-код нажатой клавиши с помощью функции inb() (определена в заголовочном файле asm/io.h), одна работа выводит скан-код клавиши в буфер ядра, другая - блокируется с помощью функции msleep().

Листинг 1: Код программы my_workqueue.c

```
1  #include <linux/module.h>
2  #include <linux/workqueue.h>
3  #include <linux/interrupt.h>
4  #include <asm/io.h> //inb
5  #include <linux/delay.h> //msleep
6
7  #define KEYBOARD_IRQ 1          //IRQ number for a keyboard
       (i8042)
8  #define KBD_DATA_REG 0x60       //I/O port for keyboard
     data
9  #define KBD_SCANCODE_MASK 0x7f
10 #define KBD_STATUS_MASK 0x80
11
12 static int counter = 0;
13 static int my_dev_id;
14 static char scancode;
15
16 MODULE_LICENSE("GPL");
17 MODULE_AUTHOR("Anastasiia Namestnik");
18
19 static struct workqueue_struct *my_wq; //workqueue
20
21 static void my_wq_function1(struct work_struct *work);
22 static void my_wq_function2(struct work_struct *work);
23
24 DECLARE_WORK(my_work1, my_wq_function1);
25 DECLARE_WORK(my_work2, my_wq_function2);
26
27 //Keyboard interrupt handler
28 //When any key is pressed keyboard controller recognises it
       and
29 //sends its scan code to a port 60h
```

```
30 irqreturn_t irq_handler(int irq, void *dev_id)
31 {
32     if (irq == KEYBOARD_IRQ)
33     {
34         ++counter;
35         printk(KERN_INFO "Workqueue: my IRQ handler was
               called %d times\n", counter);
36
37         scancode = inb(KBD_DATA_REG);
38
39         //Add work to the workqueue
40         printk(KERN_INFO "Workqueue: Queue the first work\n
               \n");
41         queue_work(my_wq, &my_work1);
42
43         printk(KERN_INFO "Workqueue: Queue the second work\
               n\n");
44         queue_work(my_wq, &my_work2);
45
46         return IRQ_HANDLED;
47     }
48     else
49         return IRQ_NONE;
50
51 }
52
53
54 static void my_wq_function1(struct work_struct *work)
55 {
56   printk(KERN_INFO "Workqueue_Work1: Scan Code %x %s\n",
57         scancode & KBD_SCANCODE_MASK,
58           scancode & KBD_STATUS_MASK ? "Released" : "
                 Pressed");
59
60    return;
61 }
62
63
64 static void my_wq_function2(struct work_struct *work)
65 {
```

```
66      printk(KERN_INFO "Workqueue_Work2: starts sleeping");
67      //msleep - sleep safely even with waitqueue
            interruptions
68       msleep(10);
69      printk(KERN_INFO "Workqueue_Work2: stops sleeping");
70
71    return;
72  }
73
74
75  static int __init myworkqueue_module_init(void)
76  {
77    printk(KERN_INFO "myworkqueue_module: Loading module...\n
        ");
78
79    if (request_irq(KEYBOARD_IRQ, irq_handler, IRQF_SHARED, "
        keyboard", &my_dev_id))
80      {
81          printk(KERN_ERR "Workqueue: Error on request_irq\n"
                );
82          return -ENOMEM;
83      }
84
85    //workqueue creation
86    if ((my_wq = alloc_workqueue("my_queue", WQ_UNBOUND, 2)))
          {
87      printk(KERN_INFO "Workqueue was successfully created!\n
          ");
88    }
89    else
90    {
91      free_irq(KEYBOARD_IRQ, (void *)(irq_handler));
92      printk(KERN_ERR "Workqueue was not created\n");
93      return -ENOMEM;
94    }
95
96    printk(KERN_INFO "myworkqueue_module: Module is now
        loaded\n");
97      return 0;
98  }
```

```
99
100
101 static void __exit myworkqueue_module_exit(void)
102 {
103     flush_workqueue(my_wq);
104     destroy_workqueue(my_wq);
105     printk(KERN_INFO "Workqueue: workqueue was destroyed\n"
        );
106     free_irq(KEYBOARD_IRQ, &my_dev_id);
107     printk(KERN_INFO "Workqueue: my IRQ handler was removed
        \n");
108     printk(KERN_INFO "Workqueue: Module is now unloaded\n")
        ;
109 }
110
111 module_init(myworkqueue_module_init);
112 module_exit(myworkqueue_module_exit);
```

На рисунке 1 приведен результат работы программы.

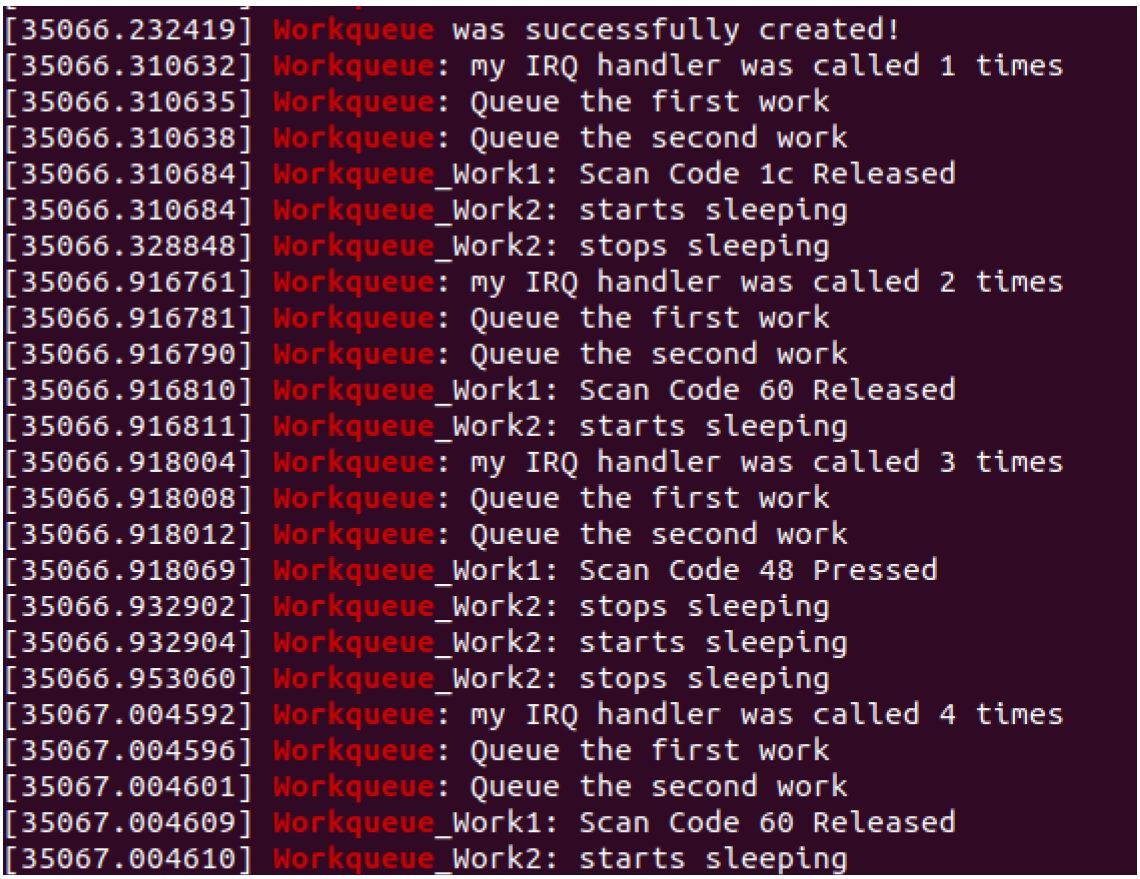

Рис 1: Результат работы программы