Отчёт по лабораторной работе №7 по дисциплине GNU/Linux

Андрей Бареков

December 6, 2019

1 Цель работы

Реализовать драйвер простого устройства в виде модуля ядра и подключить его на уровне ядра.

2 Задачи

1. Ознакомиться с представленным материалом.
2. Исправить ошибки.
3. Скомпилировать драйвер как модуль ядра.
4. Подключить драйвер как модуль на уровне ядра.
5. Продемонстрировать корректную работу драйвера.

3 Ход работы

3.1 Исправление ошибок

Удалён макрос **PREPARE WORK**. У **INIT WORK** изменилась сигнатура, поэтому в эту функцию передается структура.

```
irqreturn_t irq_handler(int irq, void *dev_id, struct
pt_regs *regs)
{
  static int initialised = 0;
  static unsigned char scancode;
  static struct work_struct task;
  unsigned char status;

  status = inb(0x64);
  scancode = inb(0x60);

  if (initialised == 0) {
    INIT_WORK(&task, got_char, &scancode);
    initialised = 1;
  } else {
    PREPARE_WORK(&task, got_char, &scancode);
  }

  queue_work(my_workqueue, &task);

  return IRQ_HANDLED;
}

```

```
irqreturn_t irq_handler(int irq, void *dev_id, struct
pt_regs *regs)
{
  static struct work_struct task;
  unsigned char status;
  static unsigned char scancode;

  status = inb(0x64);
  scancode = inb(0x60);
  got_char(&scancode);

  INIT_WORK(&task, (work_func_t)got_char);
  queue_work(my_workqueue, &task);

  return IRQ_HANDLED;
}

```

Изменилось название флага **SA SHIRQ** на **IRQF SHARED**.

```
return request_irq(1, irq_handler, SA_SHIRQ,
  "test_keyboard_irq_handler", (void *)(irq_handler));

```

```
    return request_irq(1, irq_handler, IRQF_SHARED,
      "test_keyboard_irq_handler", (void *)(irq_handler));

```

3.2 Компиляция

В папке с исходным кодом модуля создается **Makefile** для сборки модуля:

```
    obj-m += interrupt-handler.o

```

Компилируется драйвер как модуль ядра, приведен пример для Arch-подобных систем:

```
    sudo make -C /lib/modules/‘uname -r‘/build M=$PWD modules

```

3.3 Подключение драйвера как модуля на уровне ядра

Драйвер подключается командой:

```
    sudo insmod interrupt-handler.ko

```

Чтобы не повредить файловую систему, открывается новый эмулятор терминала и вводится команда, которая перезагрузит систему через определенный отрезок времени:

```
    shutdown -r 120

```

3.4 Проверка корректности работы модуля

После нажатия клавиш, в логи ядра записываются их скан коды. Открывается файл логгирования ядра:

```
    sudo dmesg

```

И показывается корректность работы драйвера:

```
    Scan Code 34 Pressed.
    Scan Code 34 Released.
    Scan Code 56 Pressed.
    Scan Code 56 Released.
    Scan Code 71 Pressed.
    Scan Code 71 Released.

```

4 Выводы

Был реализован и подключен драйвер в виде модуля на уровне ядра. В процессе достижения данной цели, были исправлены ошибки в коде обработчика прерываний, скомпилирован, подключен драйвер и показана его корректная работа.