Кто удалял загадки:

1. Nick Pentkovsky

Лаба с буфферизированным IO:

**Зачем супер блок?**

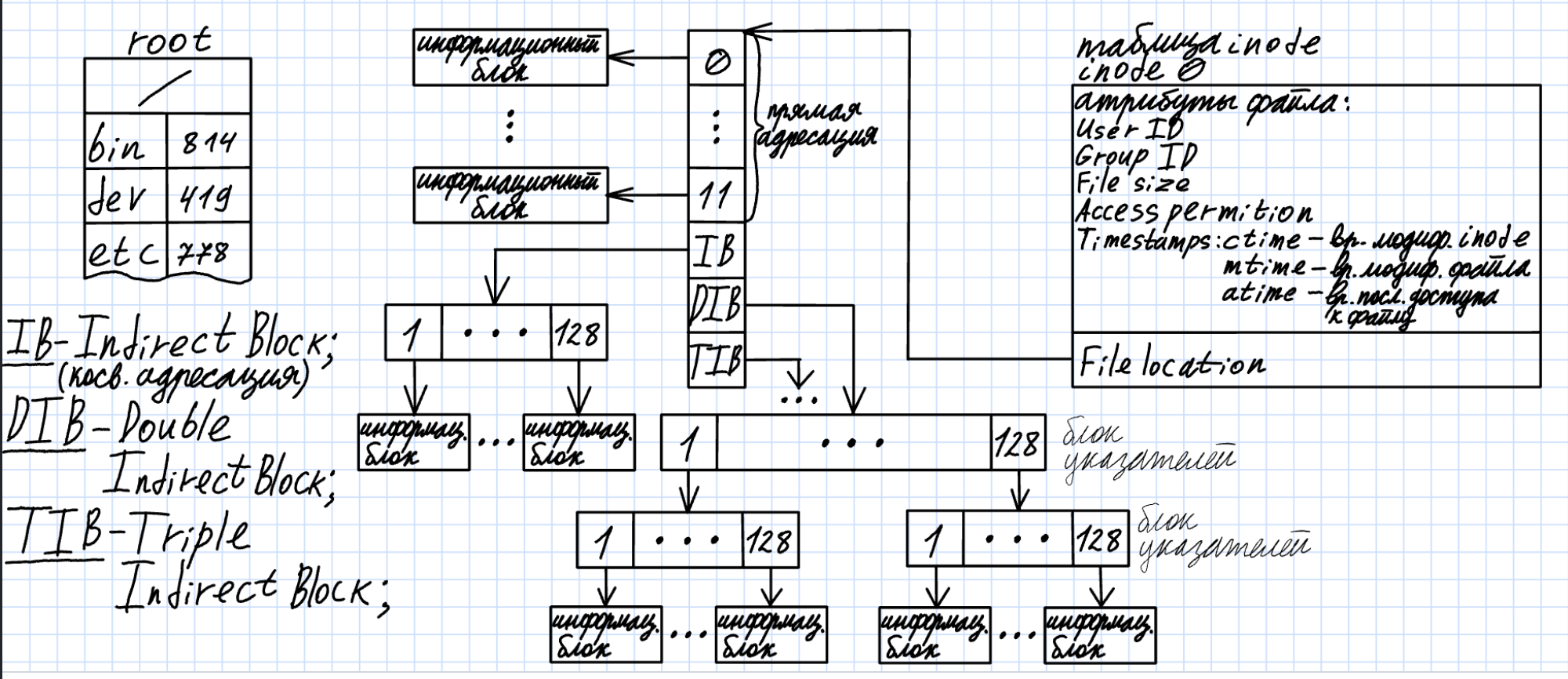
Суперблок содержит информацию о файлах в файловой системе в том числе inodов

Ну, каждому айноду соответствует файл или директория

Вообще суперблок описывает **подмонтированную** ФС

ну суперблок еще содержит список айнодов, по айноду мы можем получить информационный блок (блок на диске, он же физичкеский)

Вот так вот выглядят блоки:



**Почему в 3 пишутся только нечетные**

Fd1 fd2 раздельные структуры, разные указатели f\_pos

**Почему такой результат с fopen**

запись из буфера происходит при вызове fclose

**Чем отличаются 2 и 3 программы?**  
1. там где опен:

у каждого дескриптора открытого файла свой f\_pos, и каждая операция записи в файл write свдигает позицию,

разные f\_pos изменяются независимо, и один будет перезаписывать другой

**только там где опен надо поправить:**

окажутся четные символы (они записаны последней операцией write)

2. фопен:

запись из буфера в файл происходит при вызове fclose(), и в файле окажутся символы, записанные через файловый дескриптор, закрытый последним

# Лаба с такслетами и ворккью

Полезная ссылка: <https://0xax.gitbooks.io/linux-insides/content/Interrupts/linux-interrupts-9.html>

Чеклист:

1. массив ascii не называть my\_ascii,
2. В коде, в ините сначала реквест ирк, потом все остальное.Если не удалось установить реквест ирк, дальнейшие действия бессмысленные. То есть проверяем код возврата, и если он плохой, то выходим с ошибкой (?)
3. Изучить схему из первого семестра. Рязанова смотрит на нее и говорит: ​​”По линии IRQ сигнал от устройства приходит на контроллер прерывания”

Вопросы:

**В чем отличие такслетов, SoftIRQ, и ворккью?**

Тасклеты это надстройка над SoftIRQ.

SoftIRQ определен в системе, чтобы добавить новый, нужно перекомпилировать ядро.

Тасклет не может блокироваться, т.к. выполняется в контексте прерывания.

Тасклет привязан к прерываниям, ворк не привязан к ним и выполняется в контексте потока ядра.

Ворки выполняются в контексте процесса, что позволяет им блокироваться, уходить в сон.

Тасклет не сериализуется, в отличии от очереди. Следовательно тасклет может выполняться только на одном процессе. Неделимо.

SoftIRQ могут выполняться параллельно, в отличае от такслетов, которые поверх него. Но еще SoftIRQ не блокируются как и тасклеты.

1. Workqueue functions run in the context of a kernel process, but tasklet functions run in the software interrupt context.
2. Tasklets always run on the processor from which they were originally submitted
3. This means that workqueue functions must not be atomic as tasklet function

Сериализация - это про то что тасклеты одного типа не могут выполняться на разных процессорах. Не про перевод тасклетов в джейсоны, а про серийное (serial) выполнение.

**Что делаете в обработчике прерывания?**

1. Получаю код нажатой клавиши
2. Устанавливаю его в аргумент тасклета
3. Добавляю тасклет в планировщик (для тасклета)

Да, scheduler переводится как планировщик :)

**Что делаете в работах work queue?**

В одной - блокируем.

В другой - выводим имя клавиши

**Кто выполняет softirq?**

Eго выполняет в **контексте прерывания** демон ksoftirqd

**Какие действия и в каких местах кода, нужно выполнить, чтобы запланировать выполнение ворков?**

Аллоцируется очередь работ, структура workqueue\_struct

Инициализируются ворки - используется макрос INIT\_WORK(work\_struct)

Ворки добавляются в очередь с помощью функции queue\_work(workqueue\_struct, work\_struct)

**Что за 1 в /proc/interrupts?**

Это номер линии в обработчике прерываний. 1 соответствует клавиатуре PS/2.

**Какие действия нужно выполнить в коде чтобы зарегистрировать свой обработчик прерываний?**

Нужно определить функцию обработчика прерываний.

Потом нужно вызвать ret = request\_irq(KEYBOARD\_IRQ, my\_irq\_handler, IRQF\_SHARED, "test\_my\_irq\_handler", (void \*) my\_irq\_handler);

Аргументы:

1. Номер линии irq.
2. Имя функции - address.
3. Далее flags, у нас один flag - shared.
4. Название
5. dev\_id

Регистрируем прерывание (IRQ\_NUM=1 - клавиатура, my\_handler - указатель на обработчик прерывания

// IRQF\_SHARED - разрешает разделение irq несколькими устройствами,

// "test\_my\_irq\_handler" - имя устройства, вызывающего прерывание (для справки),

// my\_handler - id девайса, нужно при удалении. Если больше одного обработчика, то при удалении обязательно указывать, какой обработчик удаляем. Технически это id обработчика.

Так как у нас один обработчик, то можно вписывать туда NULL. Это назвали dev\_id, тк подразумевается, что у одного девайса один обработчик прерываний.)

**Зачем нужен dev\_id в request\_irq?**

данные по указателю dev требуются для удаления устройства

**Как мы регистрируем наши тасклеты и ворки? Статически или динамически? В чем разница статической и динамической регистрации?**

Динамически.

статически - во время инициализации

динамически - во время выполнения программы.

**Что такое Тасклет?**

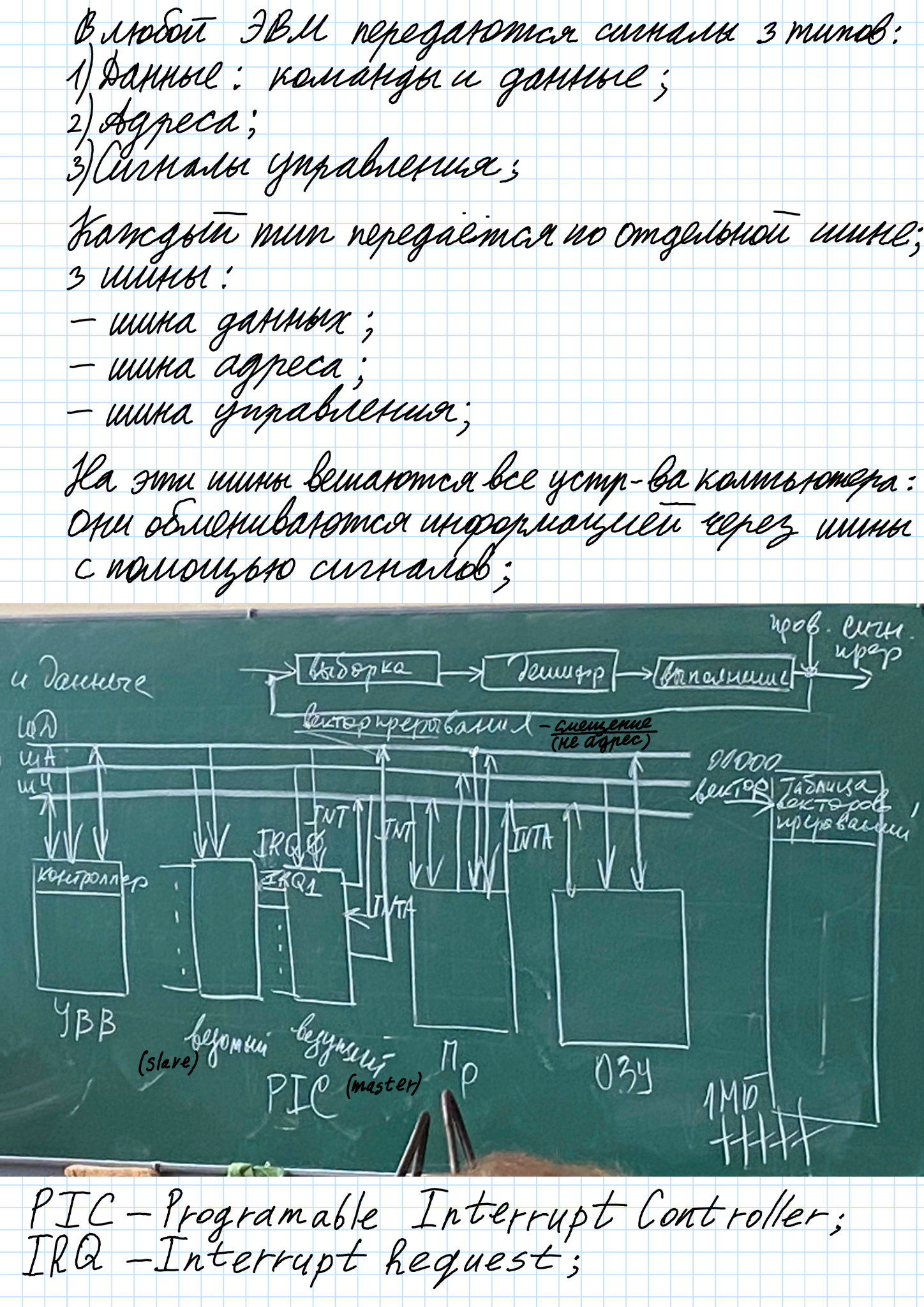
тасклет - отложенное действие

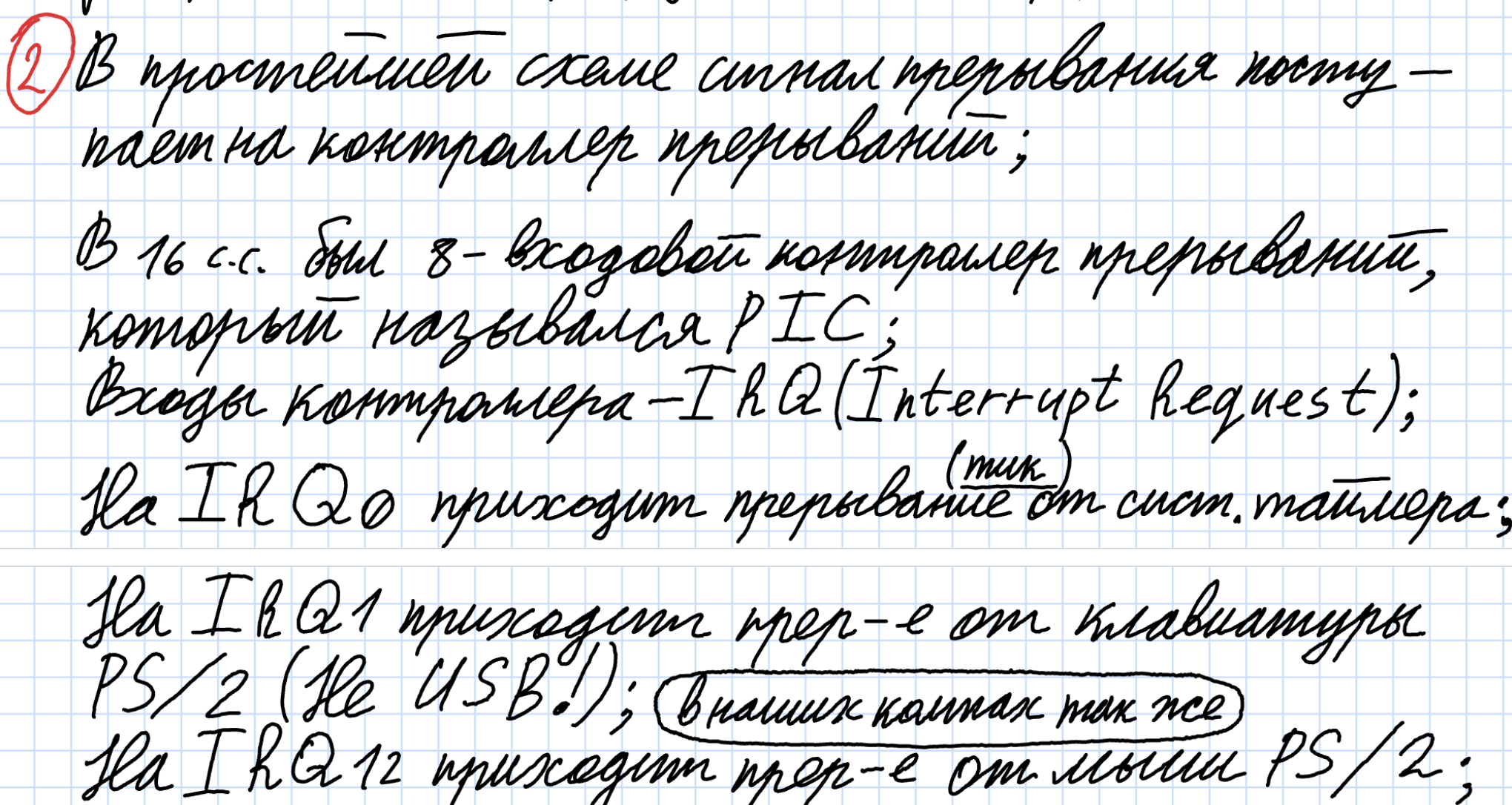
**Где посмотреть системные интеррапты с proc?**

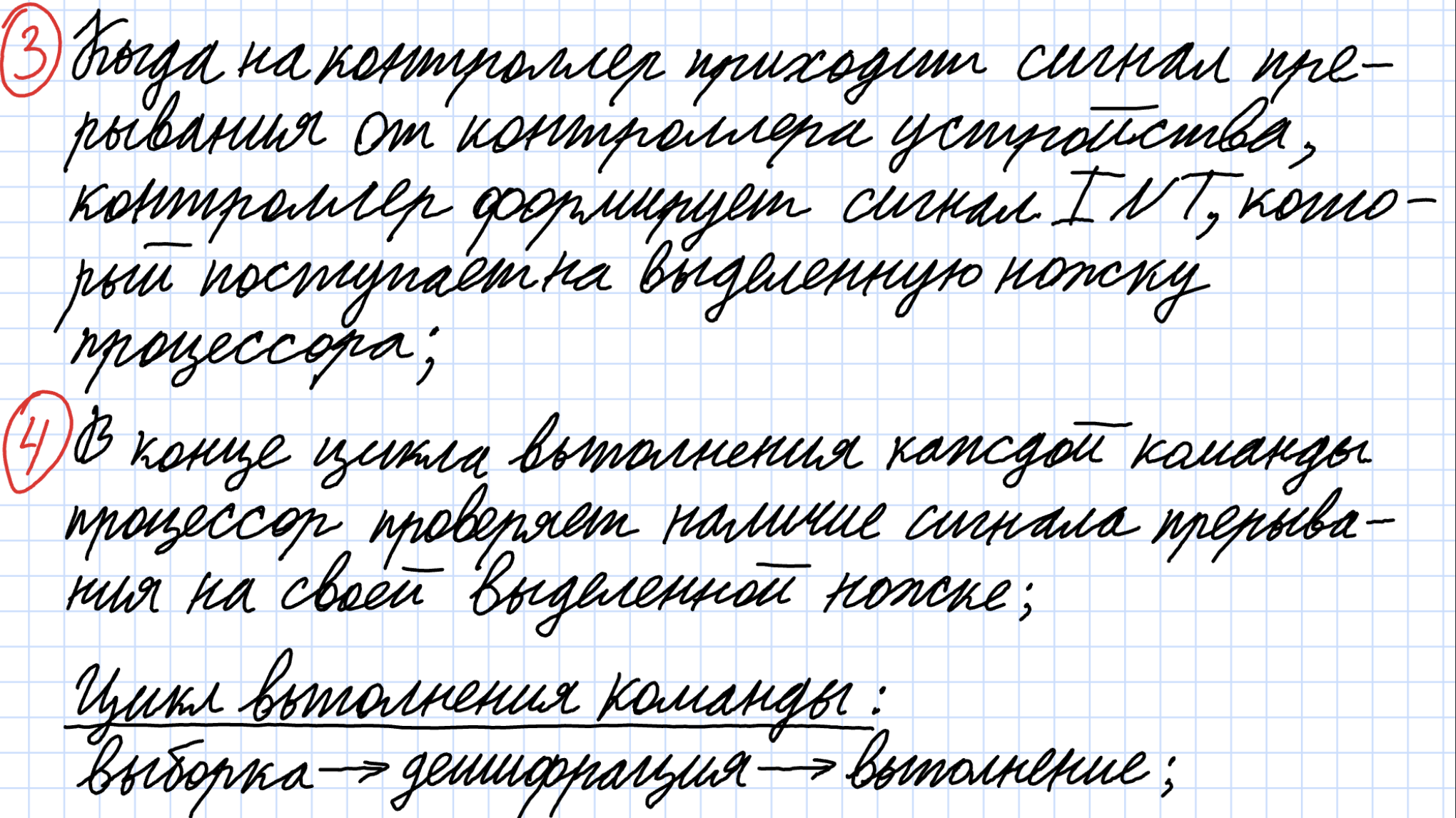
cat /proc/interrupts

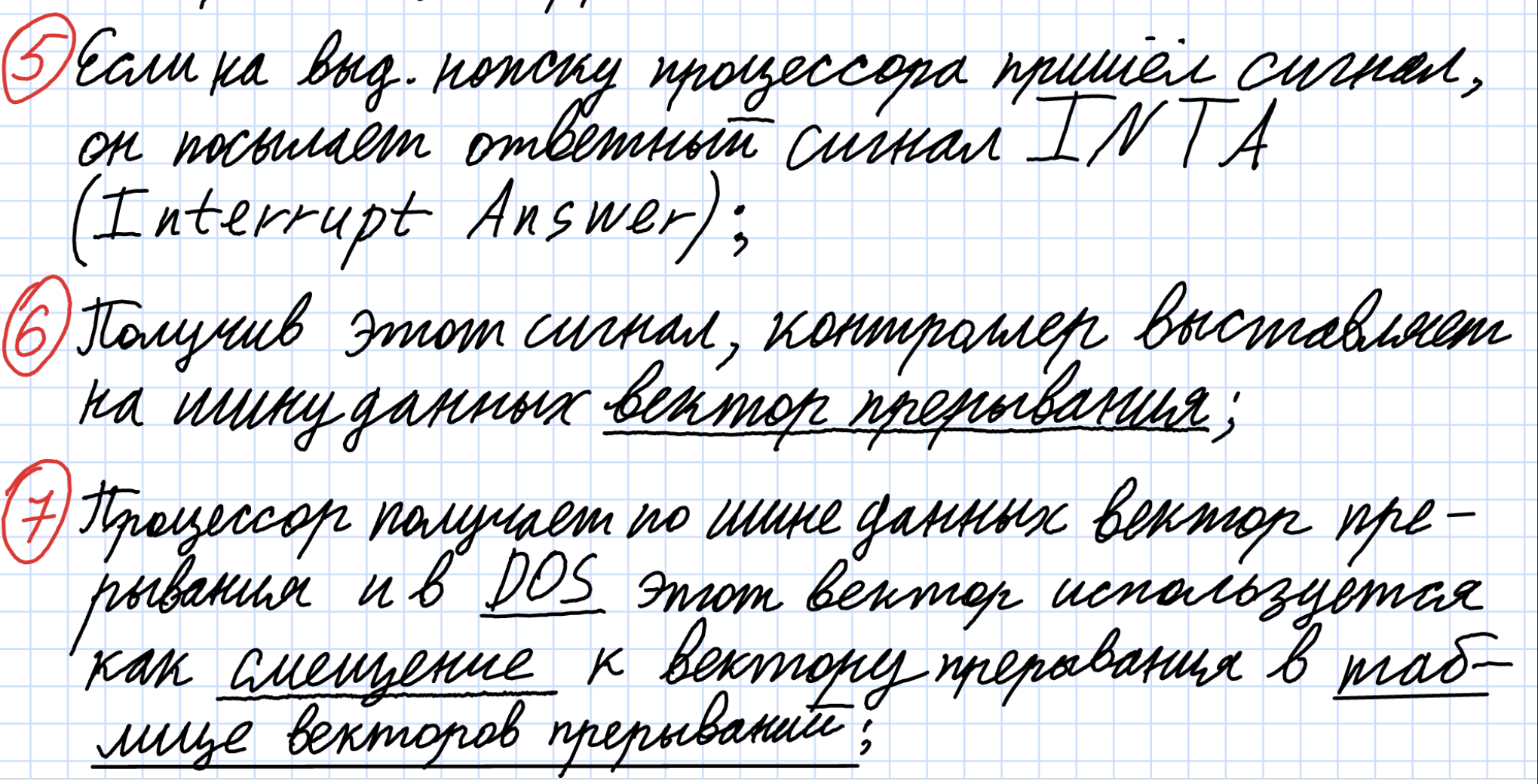
**В какую очередь добавляем ворк?**

В свою, которую аллоцировали

**Трехшинная архитектура:**







**Какие действия и в каких местах кода, нужно выполнить, чтобы запланировать выполнение ворков**

1. зарегистрировать прерывание с функцией request\_irq

2. создать очередь, тип work\_queue\_struct

3. инициализировать работы-ворки, динамически, то есть во время выполнения программы

4. В хендлере получаем код клавиши и с помощью queue\_work

5. Их выполнит поток ядра, системный поток kworker

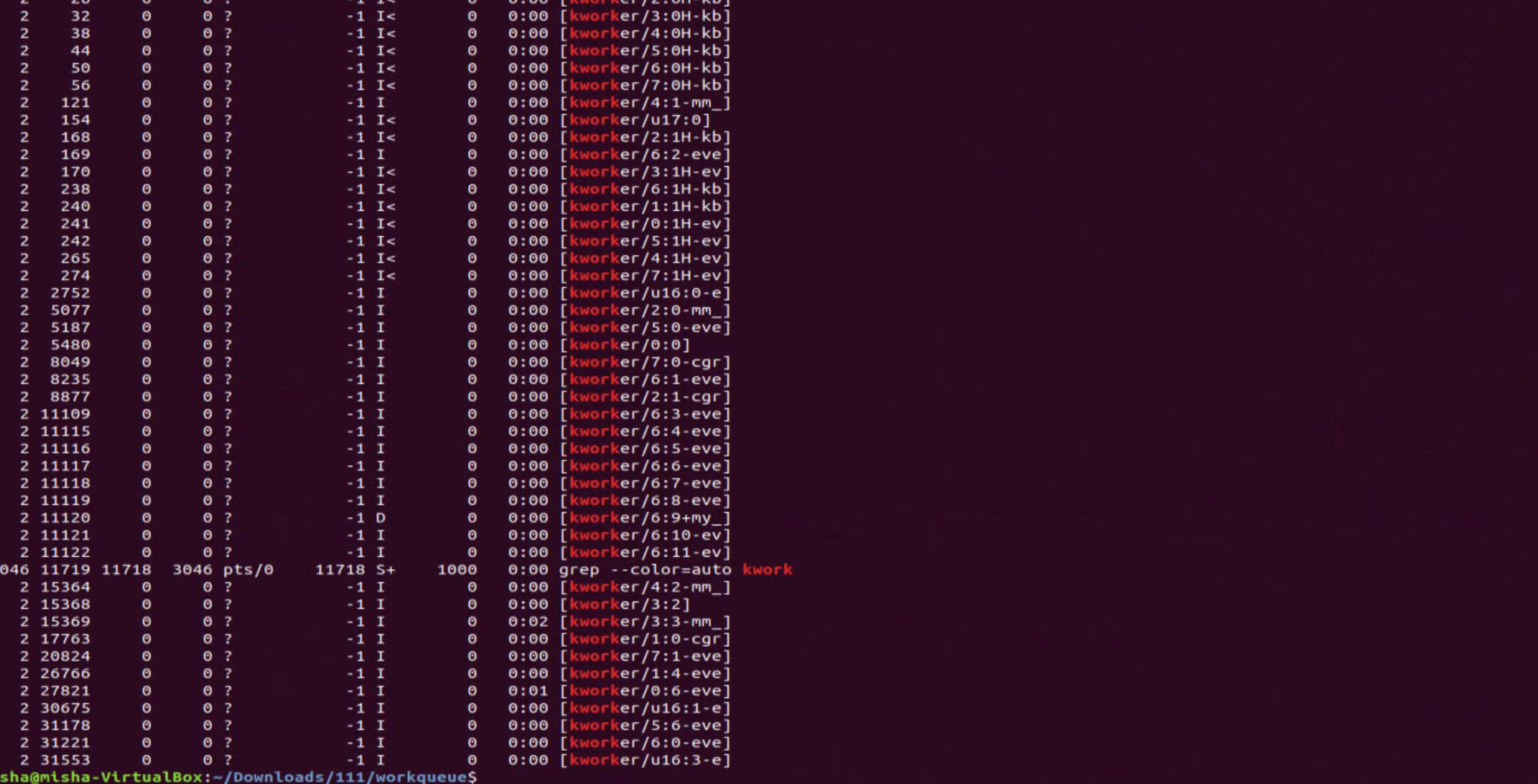
**Какая система была на лекции?**

Сетевая

**Покажите, что ваш ворк кью имеет очередь, где она?**

Загружаем модуль с очередью, далее

$ ps -jax | grep kworker

****

Вот он, начинается с +my\_