СВД
Встраиваемые
Системы

Применение инструмента TraceKev для исследования взаимодействия прикладных и системных компонентов ЗОСРВ «Нейтрино»

Усов Дмитрий

Разработчик
Отдел инструментальных средств

TraceKev – инструмент анализа трасс

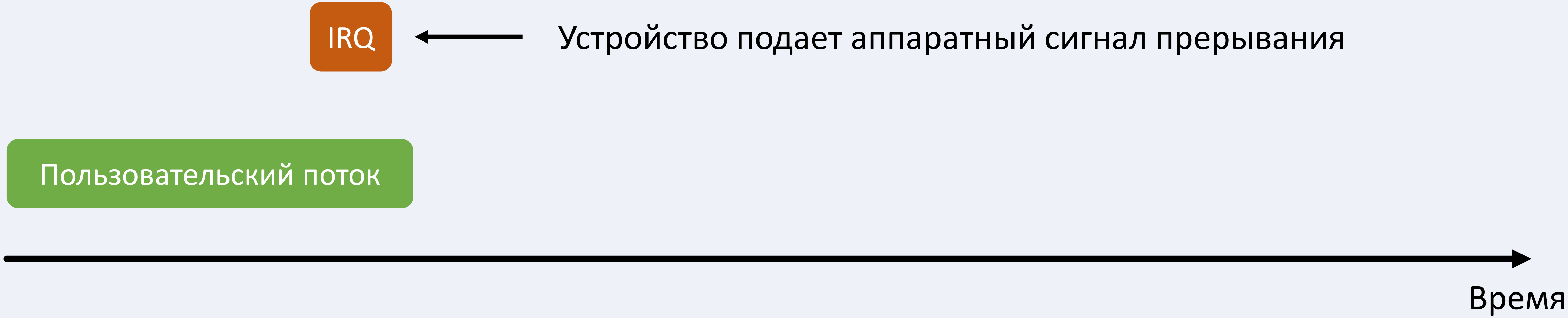
- **Для чего?**
  - TraceKev позволяет упростить анализ системных трасс, собранных на узлах под управлением ЗОСРВ «Нейтрино» для дальнейшей диагностики, отладки и поиска ошибок в работе приложений или системы
- **Примеры использования**
  - Анализ характеристик реального времени приложений
  - Исследование запуска системы
  - Изучение межпроцессного взаимодействия программ (IPC)
  - Нахождение ошибок или аномального поведения (задержки прерываний, ошибки в многопоточных приложениях, нарушения обмена сообщениями)

Что такое задержка прерывания?

- **Задержка прерывания (Interrupt Latency)**
  - это время, которое проходит от момента возникновения запроса прерывания (например, от устройства или таймера) до начала выполнения обработчика прерывания (**ISR**).

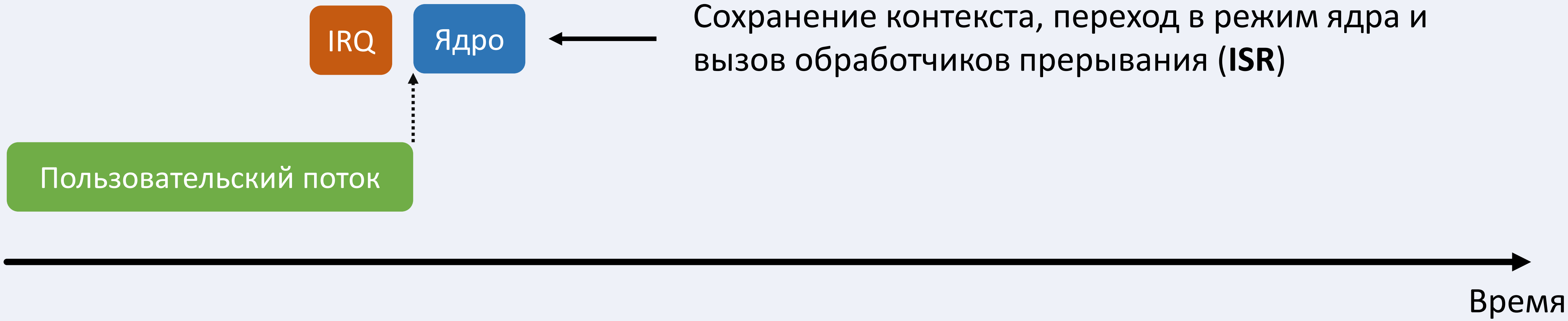
Что такое задержка прерывания?

- **Задержка прерывания (Interrupt Latency)**
  - это время, которое проходит от момента возникновения запроса прерывания (например, от устройства или таймера) до начала выполнения обработчика прерывания (**ISR**).



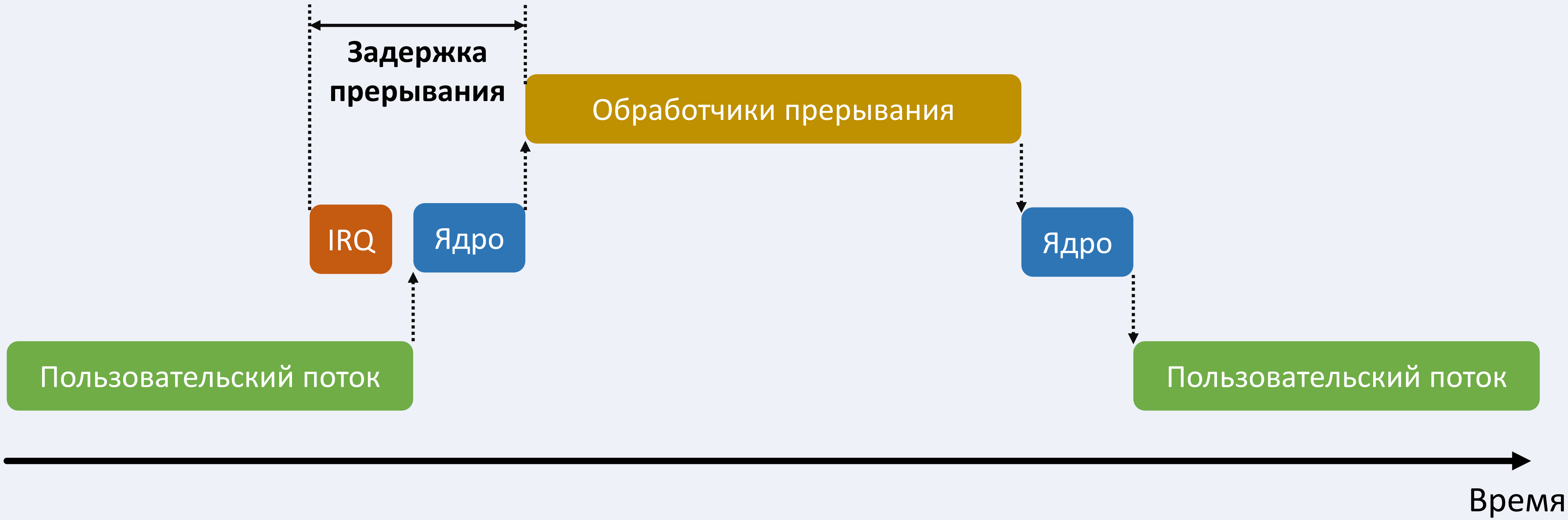
Что такое задержка прерывания?

- **Задержка прерывания (Interrupt Latency)**
  - это время, которое проходит от момента возникновения запроса прерывания (например, от устройства или таймера) до начала выполнения обработчика прерывания (**ISR**).



Что такое задержка прерывания?

- **Задержка прерывания (Interrupt Latency)**
  - это время, которое проходит от момента возникновения запроса прерывания (например, от устройства или таймера) до начала выполнения обработчика прерывания (**ISR**).



Задержка прерывания: проблема (пример)

- **Описание проблемы**
  - У нас есть обработчик прерывания, привязанный к аппаратному прерыванию (прерывание таймера **0x0**);
  - Наши измерения показывают, что обработчик имеет неприемлемую задержку при обработке прерывания;
  - Как мы можем определить, что конкретно в системе вызывает эту проблему?
- **Необходимо проанализировать причину задержки прерывания**
  - Попытаемся разобраться с этим, вплоть до строки исходного кода, вызывающей проблему

Задержка прерывания: пример

- **Исполняемые файлы**
  - В нашем примере будут использоваться два отдельных исходных файла:
    - **mass_driver.c** – программа-драйвер, обрабатывающая прерывания;
    - **hw_interrupt** – программа, определяющая большую задержку прерывания.

Задержка прерывания: пример

- **Часть hw_interrupt.c:**

```c
const sigevent *interrupt_handler(void *area, int id) {
    clocks = ClockCycles();
    count++;

    delta       = clocks - last_clocks;
    last_clocks = clocks;

    if (delta > too_many_clocks) {
        sdelta = delta;
        return &int_event;
    } else {
        return nullptr;
    }
}
```

Обработчик прерывания сам обнаруживает, если он столкнулся с большой задержкой прерывания.

- **Определение задержки**
  - В обработчике прерываний, мы проверяем счетчик времени. Если время, прошедшее между двумя прерываниями превышает рассчитанное время, то мы уведомляем главный поток

Задержка прерывания: пример

- **Часть hw_interrupt.c:**

```
int main(int argc, char **argv) {
    ...

    while (true) {
        // Блокируемся здесь, ожидая уведомления от нашего обработчика прерываний
        InterruptWait(0, nullptr);

        // Формируем пользовательское событие, которое информирует о задержке
        int ret=0;
        if (ret = trace_logf(_NTO_TRACE_USERFIRST + 1, "hw_interrupt: Too great irq latency"); ret == -1) {
            std::cout << ret << std::endl;
            error_out("trace_logf", errno);
        }

```

- **Пользовательское событие**
  - Если пришло уведомление от нашего обработчика прерываний, то значит возникла большая задержка;
  - Формируем пользовательское событие, записываемое в трассу событий

Задержка прерывания: запись трассы

- **Для вызова проблемы и создания трассы событий:**
  - Запустим скомпилированные программы из примера на целевой системе:
    - **hw_interrupt:**

```
# ./hw_interrupt
```

    - **mass_driver:**

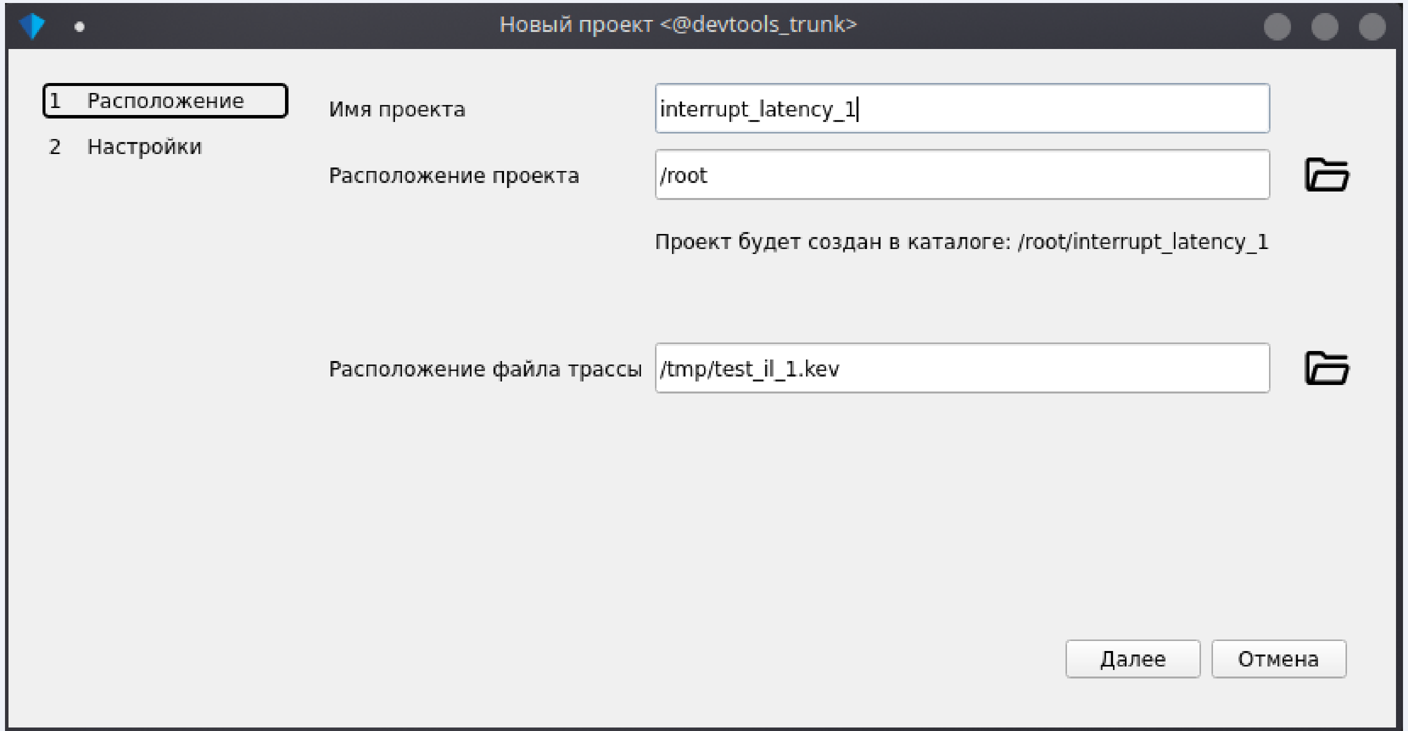
```
# ./mass_driver  -i 0
```

  - Записываем трассу системных событий с помощью программы **tracelogger:**

```
# tracelogger -s 4 -f ./test_il_1.kev
```

Задержка прерывания: определение проблемы

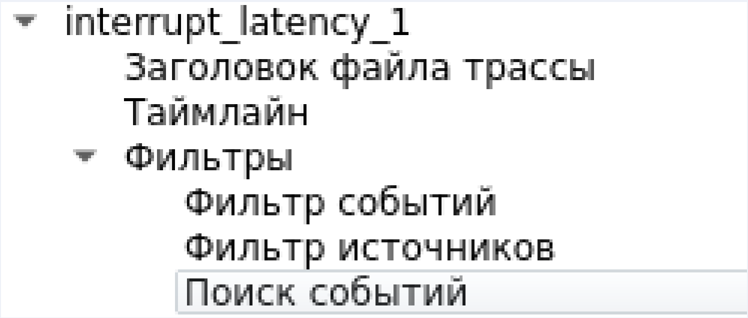
- После записи трассы системных событий необходимо перенести файл **test_il_1.kev** на инструментальную систему
- Запустить приложение для анализа системных трасс «**TraceKev**», входящее в Комплект разработчика
- Создать новый проект с указанием файла собранной трассы



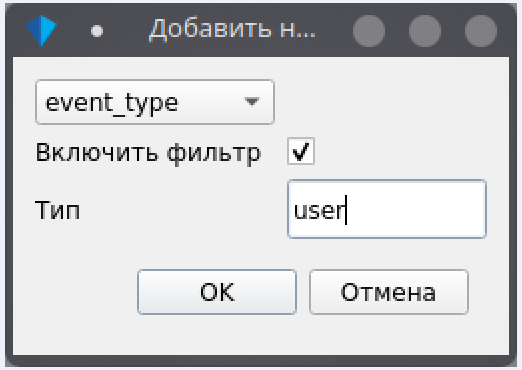
Задержка прерывания: определение проблемы

- Теперь можно найти пользовательские события, генерируемые **hw_interrupt**

Открываем инструмент «**Поиск событий**»



Добавляем фильтр поиска по типу события «**user**»



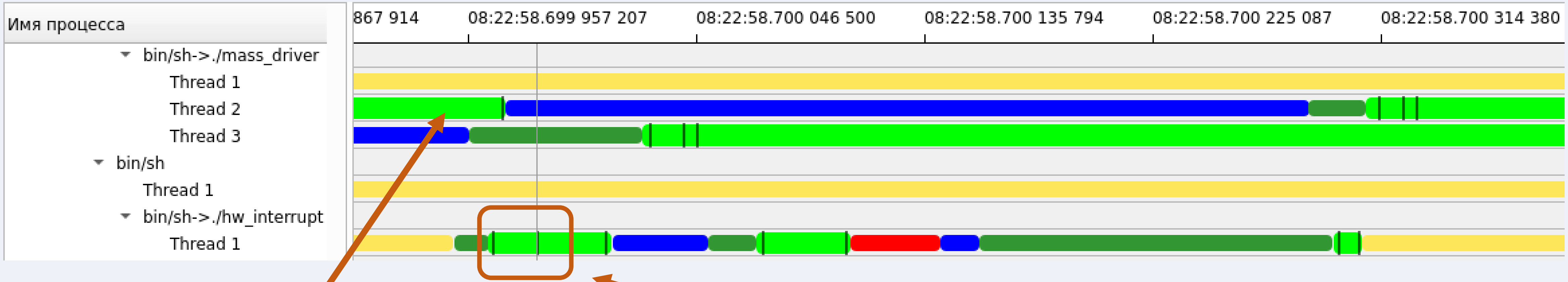
Получаем следующие события:

Результаты поиска

Метка времени	CPU №	Источник события	Тип	Параметры
08:22:58.120 430 433	1	bin/sh->./hw_interrupt->Thread 1	user	event = 1; str = hw_interrupt: Too great irq latency; d0 = 0x695f7768; d1 = ...
08:22:58.124 350 555	1	bin/sh->./hw_interrupt->Thread 1	user	event = 1; str = hw_interrupt: Too great irq latency; d0 = 0x695f7768; d1 = ...
08:22:58.128 998 262	3	bin/sh->./hw_interrupt->Thread 1	user	event = 1; str = hw_interrupt: Too great irq latency; d0 = 0x695f7768; d1 = ...
08:22:58.132 940 644	2	bin/sh->./hw_interrupt->Thread 1	user	event = 1; str = hw_interrupt: Too great irq latency; d0 = 0x695f7768; d1 = ...
08:22:58.135 258 660	2	bin/sh->./hw_interrupt->Thread 1	user	event = 1; str = hw_interrupt: Too great irq latency; d0 = 0x695f7768; d1 = ...
08:22:58.173 587 804	2	bin/sh->./hw_interrupt->Thread 1	user	event = 1; str = hw_interrupt: Too great irq latency; d0 = 0x695f7768; d1 = ...

Задержка прерывания: определение проблемы

- Теперь рассмотрим подробнее возникновение данного события на **таймлайне:**



mass_driver был запущен

hw_interrupt сообщил о большой задержки прерывания

Задержка прерывания: определение проблемы

- **Мы выявили процесс, вызывающий проблему, однако его функциональное назначение пока неизвестно**
  - Чтобы понять, как происходит выполнение программы, необходимо отобразить последовательность её работы;
  - Для этого следует добавить пользовательские события трассировки в различные участки кода, которые выполняются в момент возникновения задержек;
  - В частности, нужно раскомментировать вызовы функций *trace_log*()* в исходном файле **mass_driver.c**;
  - Теперь можно скомпилировать программу **mass_driver** и повторно собрать трассу с помощью утилиты **tracelogger**.

Задержка прерывания: определение проблемы

- Исходный код **mass_driver** теперь будет содержать строки с добавлением пользовательских событий:

```
void prep_work() {
    trace_logf(_NTO_TRACE_USERFIRST + 2, "Starting prepare work");
    ...
}

void critical_work() {
    trace_logf(_NTO_TRACE_USERFIRST + 2, "Starting critical work");
    ...
}

void begin_critical_section() {
    trace_logf(_NTO_TRACE_USERFIRST + 2, "Enter critical section");
    ...
}

void end_critical_section() {
    ...
    trace_logf(_NTO_TRACE_USERFIRST + 2, "Leave critical section");
}

void interrupt_work() {
    trace_logf(_NTO_TRACE_USERFIRST + 2, "Starting interrupt work");
    ...
}

void handle_msg(int rcvid, msgs *rcv_buf) {
    trace_logf(_NTO_TRACE_USERFIRST + 2, "Message handling");
    ...
}
```

Задержка прерывания: определение проблемы

- В процесс, вызывающий задержку, были добавлены дополнительные пользовательские события – это позволит отследить последовательность его выполнения;
- Далее необходимо найти в новой системной трассе добавленные события;
- Используя данные этих событий, определим, какая последовательность действий привела к возникновению проблемы.

Задержка прерывания: определение причины

- После получения трассы системных событий открываем ее в TraceKev;
- Проверим, может ли проблема быть связана с ошибками в работе с **IPC;**

12:08:39.661 660 693	0	Interrupt handler 0x0 pid: 1318947	int_handler_enter	pid = 1318947; interruptNumber = 0x0; ip = 0x0; area = 0xefc8f520
12:08:39.661 663 415	0	Interrupt handler 0x0 pid: 1318947	int_handler_exit	interruptNumber = 0x0; sigEvent = INTR EVENT
12:08:39.661 664 167	0	Interrupt 0x0	int_exit	interruptNumber = 0x0; kernelFlag = 0x3000001
12:08:39.661 671 715	0	bin/sh->./mass_driver->Thread 2	thread_ready	pid = 1318947; tid = 2
12:08:39.661 682 534	1	bin/sh->./mass_driver->Thread 2	thread_running	pid = 1318947; tid = 2
12:08:39.661 687 173	1	bin/sh->./mass_driver->Thread 2	communication_rec_pulse	scoid = 0x40000003; pid = 1318947
12:08:39.661 687 654	1	bin/sh->./mass_driver->Thread 2	ker_exit_msg_receive_v	rcvid = 0x0; rmsg0 = 0x0
12:08:39.661 700 849	1	bin/sh->./mass_driver->Thread 2	ker_enter_interrupt_unmask	intr = 0; id = 13
12:08:39.661 704 985	1	bin/sh->./mass_driver->Thread 2	ker_exit_interrupt_unmask	maskLevel = 0x0

Обработчик прерывания посылает сигнал (**mass_driver**)

Задержка прерывания: определение причины

- После получения трассы системных событий открываем ее в TraceKev;
- Проверим, может ли проблема быть связана с ошибками в работе с **IPC;**

12:08:39.661 660 693	0	Interrupt handler 0x0 pid: 1318947	int_handler_enter	pid = 1318947; interruptNumber = 0x0; ip = 0x0; area = 0xefc8f520
12:08:39.661 663 415	0	Interrupt handler 0x0 pid: 1318947	int_handler_exit	interruptNumber = 0x0; sigEvent = INTR EVENT
12:08:39.661 664 167	0	Interrupt 0x0	int_exit	interruptNumber = 0x0; kernelFlag = 0x3000001
12:08:39.661 671 715	0	bin/sh->./mass_driver->Thread 2	thread_ready	pid = 1318947; tid = 2
12:08:39.661 682 534	1	bin/sh->./mass_driver->Thread 2	thread_running	pid = 1318947; tid = 2
12:08:39.661 687 173	1	bin/sh->./mass_driver->Thread 2	communication_rec_pulse	scoid = 0x40000003; pid = 1318947
12:08:39.661 687 654	1	bin/sh->./mass_driver->Thread 2	ker_exit_msg_receive_v	rcvid = 0x0; rmsg0 = 0x0
12:08:39.661 700 849	1	bin/sh->./mass_driver->Thread 2	ker_enter_interrupt_unmask	intr = 0; id = 13
12:08:39.661 704 985	1	bin/sh->./mass_driver->Thread 2	ker_exit_interrupt_unmask	maskLevel = 0x0

Поток **mass_driver** запускается и принимает сигнал

Задержка прерывания: определение причины

- После получения трассы системных событий открываем ее в TraceKev;
- Проверим, может ли проблема быть связана с ошибками в работе с **IPC;**

12:08:39.661 660 693	0	Interrupt handler 0x0 pid: 1318947	int_handler_enter	pid = 1318947; interruptNumber = 0x0; ip = 0x0; area = 0xefc8f520
12:08:39.661 663 415	0	Interrupt handler 0x0 pid: 1318947	int_handler_exit	interruptNumber = 0x0; sigEvent = INTR EVENT
12:08:39.661 664 167	0	Interrupt 0x0	int_exit	interruptNumber = 0x0; kernelFlag = 0x3000001
12:08:39.661 671 715	0	bin/sh->./mass_driver->Thread 2	thread_ready	pid = 1318947; tid = 2
12:08:39.661 682 534	1	bin/sh->./mass_driver->Thread 2	thread_running	pid = 1318947; tid = 2
12:08:39.661 687 173	1	bin/sh->./mass_driver->Thread 2	communication_rec_pulse	scoid = 0x40000003; pid = 1318947
12:08:39.661 687 654	1	bin/sh->./mass_driver->Thread 2	ker_exit_msg_receive_v	rcvid = 0x0; rmsg0 = 0x0
12:08:39.661 700 849	1	bin/sh->./mass_driver->Thread 2	ker_enter_interrupt_unmask	intr = 0; id = 13
12:08:39.661 704 985	1	bin/sh->./mass_driver->Thread 2	ker_exit_interrupt_unmask	maskLevel = 0x0

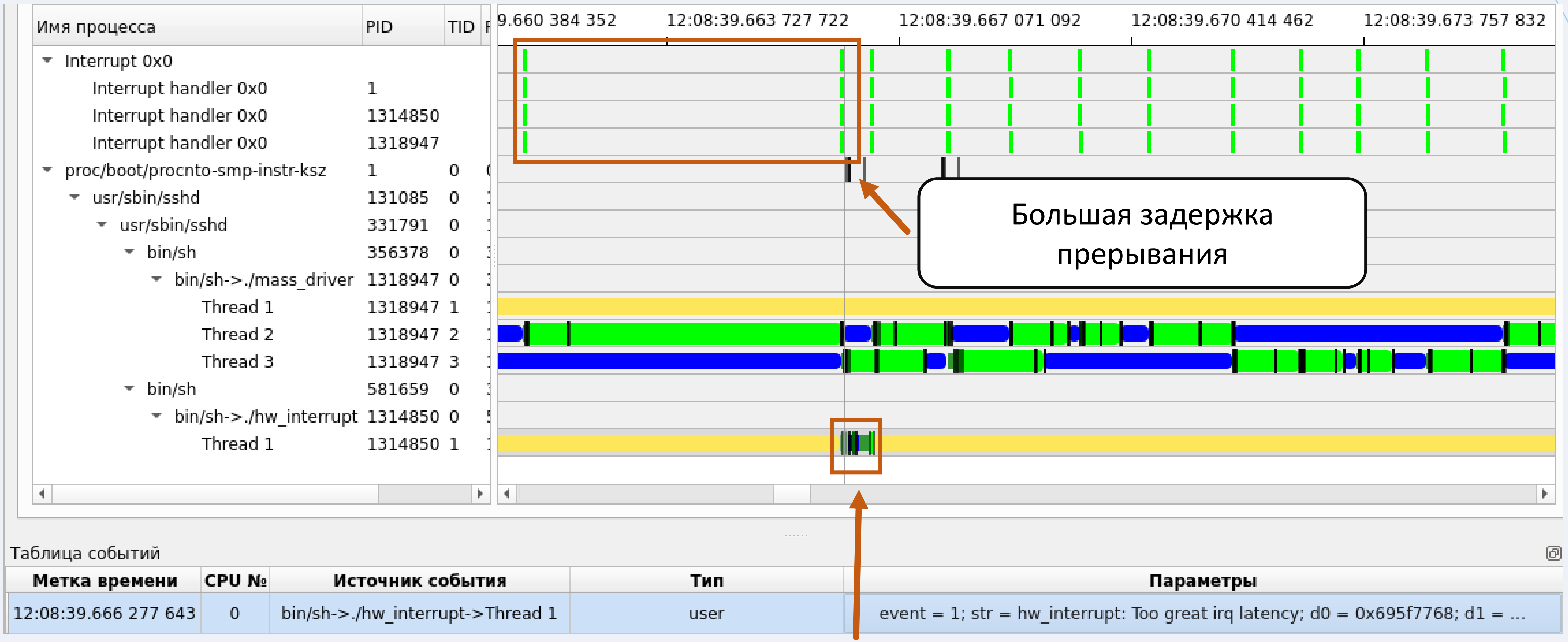
После чего размаскирует прерывание

Значит проблема в другом месте

Задержка прерывания: определение причины

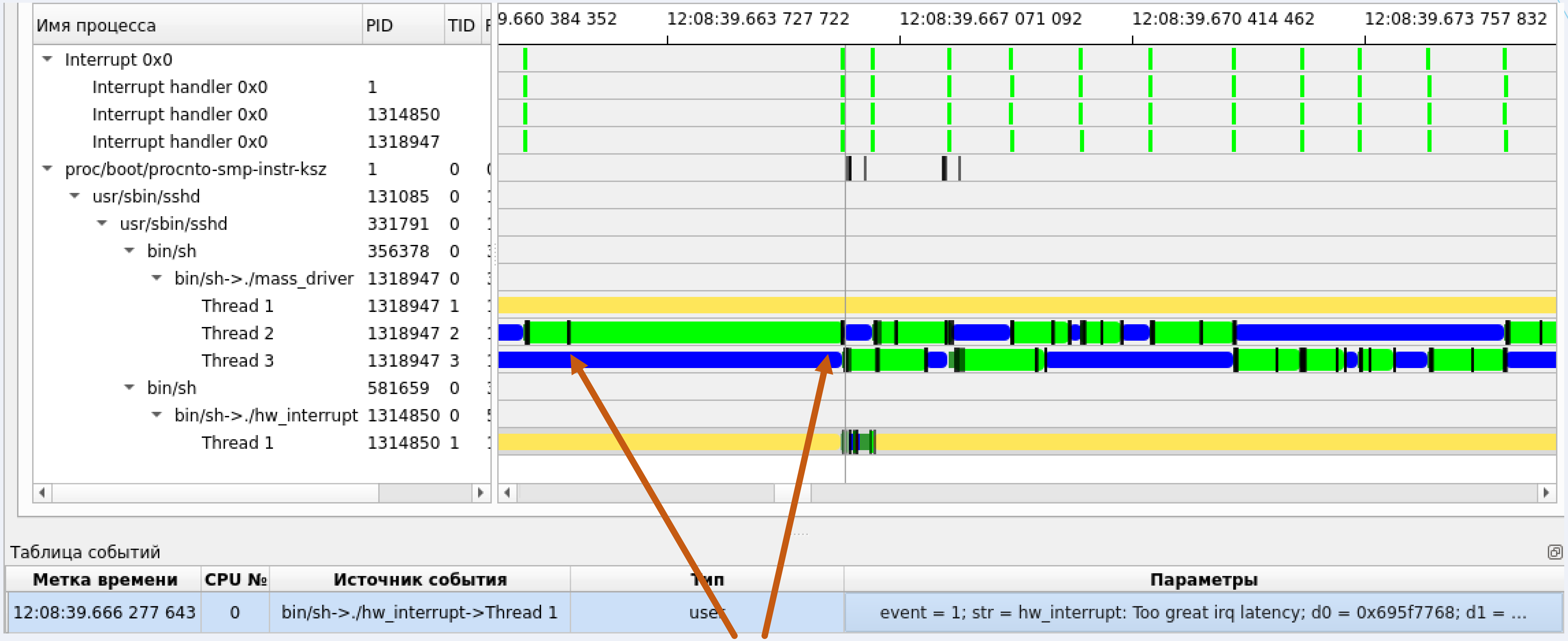
- Найдем появление пользовательского события от **hw_interrupt**;
- Смотрим по пользовательским событиям, что происходило перед возникновением проблемы.

Задержка прерывания: определение причины



Событие появляется сразу после большой задержки прерывания

Задержка прерывания: определение причины



Наблюдаем события внутри **mass_driver** в момент возникновения проблемы

Задержка прерывания: определение причины

Таблица событий

Метка времени	CPU №	Источник события	Тип	Параметры
12:41:35.096 927 574	3	bin/sh->./mass_driver->Thread 3	thread_running	pid = 1699875; tid = 3
12:41:35.096 931 920	3	bin/sh->./mass_driver->Thread 3	communication_rec_pulse	scoid = 0x40000003; pid = 1699875
12:41:35.096 932 296	3	bin/sh->./mass_driver->Thread 3	ker_exit_msg_receive_v	rcvid = 0x0; rmsg0 = 0x0
12:41:35.096 944 679	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_unmask	intr = 0; id = 13
12:41:35.096 949 062	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_unmask	maskLevel = 0x0
12:41:35.096 957 077	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting interrupt work; d0 = 0x72617453...
12:41:35.096 963 377	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting prepare work; d0 = 0x72617453; ...
12:41:35.097 248 434	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Enter critical section; d0 = 0x65746e45; d...
12:41:35.097 249 479	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_mask	intr = 0; id = 13
12:41:35.097 252 110	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_mask	maskLevel = 0x1
12:41:35.097 257 930	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting critical work; d0 = 0x72617453; d...
12:41:35.100 049 717	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Leave critical section; d0 = 0x7661654c; ...
12:41:35.100 051 612	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_unmask	intr = 0; id = 13
12:41:35.100 054 386	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_unmask	maskLevel = 0x0
12:41:35.100 055 123	3	bin/sh->./mass_driver->Thread 3	ker_enter_msg_receive_v	chid = 0x1; rparts = 1
12:41:35.100 056 153	3	b		

После получения события прерывания начинается подготовительная работа

Задержка прерывания: определение причины

Таблица событий

Метка времени	CPU №	Источник события	Тип	Параметры
12:41:35.096 927 574	3	bin/sh->./mass_driver->Thread 3	thread_running	pid = 1699875; tid = 3
12:41:35.096 931 920	3	bin/sh->./mass_driver->Thread 3	communication_rec_pulse	scoid = 0x40000003; pid = 1699875
12:41:35.096 932 296	3	bin/sh->./mass_driver->Thread 3	ker_exit_msg_receive_v	rcvid = 0x0; rmsg0 = 0x0
12:41:35.096 944 679	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_unmask	intr = 0; id = 13
12:41:35.096 949 062	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_unmask	maskLevel = 0x0
12:41:35.096 957 077	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting interrupt work; d0 = 0x72617453...
12:41:35.096 963 377	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting prepare work; d0 = 0x72617453; ...
12:41:35.097 248 434	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Enter critical section; d0 = 0x65746e45; d...
12:41:35.097 249 479	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_mask	intr = 0; id = 13
12:41:35.097 252 110	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_mask	maskLevel = 0x1
12:41:35.097 257 930	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting critical work; d0 = 0x72617453; d...
12:41:35.100 049 717	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Leave critical section; d0 = 0x7661654c; ...
12:41:35.100 051 612	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_unmask	intr = 0; id = 13
12:41:35.100 054 386	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_unmask	maskLevel = 0x0
12:41:35.100 055 123	3	bin/sh->./mass_driver->Thread 3	ker_enter_msg_receive_v	chid = 0x1; rparts = 1
12:41:35.100 056 153	3	b		

Входим в критическую секцию и маскируем прерывания

Задержка прерывания: определение причины

Таблица событий

Метка времени	CPU №	Источник события	Тип	Параметры
12:41:35.096 927 574	3	bin/sh->./mass_driver->Thread 3	thread_running	pid = 1699875; tid = 3
12:41:35.096 931 920	3	bin/sh->./mass_driver->Thread 3	communication_rec_pulse	scoid = 0x40000003; pid = 1699875
12:41:35.096 932 296	3	bin/sh->./mass_driver->Thread 3	ker_exit_msg_receive_v	rcvid = 0x0; rmsg0 = 0x0
12:41:35.096 944 679	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_unmask	intr = 0; id = 13
12:41:35.096 949 062	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_unmask	maskLevel = 0x0
12:41:35.096 957 077	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting interrupt work; d0 = 0x72617453...
12:41:35.096 963 377	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting prepare work; d0 = 0x72617453; ...
12:41:35.097 248 434	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Enter critical section; d0 = 0x65746e45; d...
12:41:35.097 249 479	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_mask	intr = 0; id = 13
12:41:35.097 252 110	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_mask	maskLevel = 0x1
12:41:35.097 257 930	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Starting critical work; d0 = 0x72617453; d...
12:41:35.100 049 717	3	bin/sh->./mass_driver->Thread 3	user	event = 2; str = Leave critical section; d0 = 0x7661654c; ...
12:41:35.100 051 612	3	bin/sh->./mass_driver->Thread 3	ker_enter_interrupt_unmask	intr = 0; id = 13
12:41:35.100 054 386	3	bin/sh->./mass_driver->Thread 3	ker_exit_interrupt_unmask	maskLevel = 0x0
12:41:35.100 055 123	3	bin/sh->./mass_driver->Thread 3	ker_enter_msg_receive_v	chid = 0x1; rparts = 1
12:41:35.100 056 153	3	b		

Выходим из критической секции и размаскируем прерывания

Задержка прерывания: определение причины

- Замерим разницу во времени между началом критической секции и выхода из нее, выделяя диапазон событий:

12:41:35.097 248 434 ⟷ 12:41:35.100 049 717 Δ: 2.801283 ms

- Время работы критической секции оказывается более 2 миллисекунд, что не допустимо

Задержка прерывания: решение проблемы

- **Как можно исправить эту проблему:**
  - Заменить маскирование и размаскирование прерываний использованием мьютексов;
    - Если маскирование используется для защиты данных между потоками (в том числе при доступе к оборудованию), то применение мьютексов будет более безопасным и корректным решением.
  - Сократить время маскирования прерываний;
    - Проверьте, действительно ли необходимо защищать весь код внутри критической секции. Возможно, часть операций можно вынести за её пределы.
  - Рассмотреть комбинированный подход.
    - Можно маскировать прерывания только на короткий промежуток времени, а затем безопасно их размаскировать, совмещая этот механизм с использованием мьютексов для защиты общих ресурсов.

СВД
Встраиваемые
Системы

Спасибо за внимание!

Усов Дмитрий

Разработчик
Отдел инструментальных средств

ул. Кузнецовская, д. 19,
г. Санкт-Петербург
+7 (812) 317-55-56
www.kpda.ru
support@kpda.ru