

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**  
«Операционные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**  
«Планировщик»

**Выполнили:**

Бардышев Артём Антонович,  
студент группы N3246

---

(подпись)

Суханкулиев Мухаммет,  
студент группы N3246



---

(подпись)

**Проверил:**

Савков Сергей Витальевич,  
инженер

---

(отметка о выполнении)

---

(подпись)

Санкт-Петербург  
2024 г.

## СОДЕРЖАНИЕ

Введение .....	3
1     Тестирование планировщиков.....	4
1.1    Скрипт для тестирования.....	4
1.2    Измерения.....	4
1.3    Анализ результатов .....	6
2     Модификация планировщика на уровне ядра.....	7
2.1    Модификация исходного кода ядра linux.....	7
2.2    Модификация существующего планировщика на уровне ядра .....	8
Заключение.....	11
Список использованных источников.....	12

## ВВЕДЕНИЕ

**Цель работы** – провести тестирование и найти лучший планировщик ввода-вывода среди других.

Усложнение:

Модифицировать существующий планировщик на уровне ядра

Планирование IO.

### Основные определения:

**Планировщик ввода-вывода (IO scheduler):** компонент операционной системы, отвечающий за управление очередями запросов на доступ к устройствам хранения данных. Он определяет порядок, в котором запросы обрабатываются, с целью повышения эффективности работы системы.

**Алгоритмы планирования:** существуют различные алгоритмы планирования ввода-вывода, каждый из которых имеет свои преимущества и недостатки:

- **CFQ (Completely Fair Queuing):** обеспечивает равный доступ к устройствам для всех процессов, стараясь минимизировать время ожидания.
- **BFQ (Budget Fair Queueing):** обеспечивает более эффективное распределение ресурсов, учитывая приоритеты процессов и их бюджет на ввод-вывод. Это позволяет улучшить производительность для задач с разными требованиями.
- **Kyber:** предназначен для работы в реальном времени и ориентирован на минимизацию задержек, что делает его подходящим для систем, требующих быстрого отклика. Kyber использует адаптивный подход к управлению запросами, обеспечивая высокую производительность.
- **None:** режим, в котором нет никакого планирования, и запросы обрабатываются в порядке их поступления, что может привести к неравномерному распределению ресурсов и высоким задержкам в случае интенсивной нагрузки.
- **mq-deadline:** планировщик, который использует подход, основанный на сроках выполнения запросов. Он минимизирует задержки, обрабатывая запросы с учетом их приоритетов и времени ожидания, что делает его эффективным для сценариев с высокой нагрузкой.

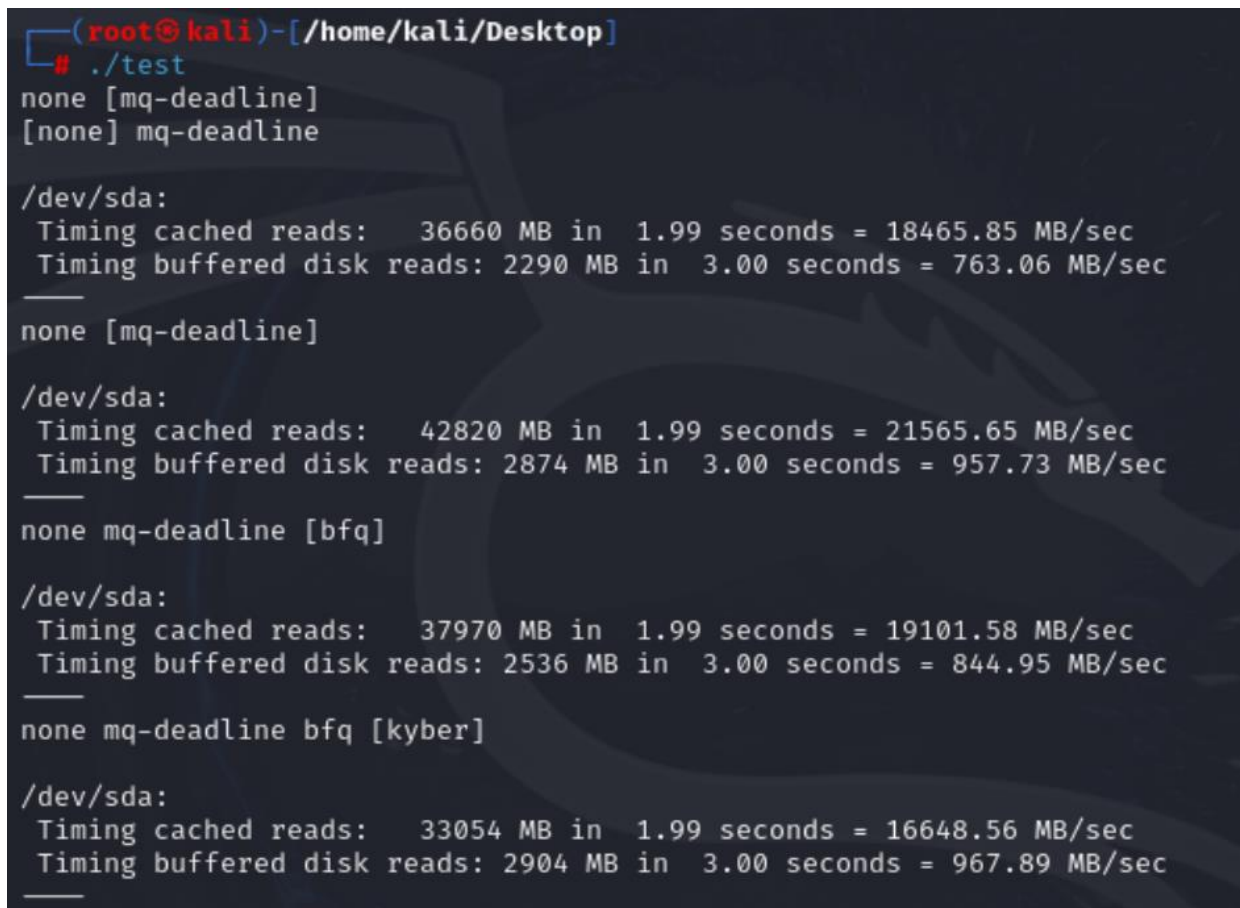
## 1 ТЕСТИРОВАНИЕ ПЛАНИРОВЩИКОВ

### 1.1 Скрипт для тестирования

```
#!/bin/bash
DISC="sda"
cat /sys/block/$DISC/queue/scheduler
for T in none mq-deadline bfq kyber; do
    echo $T > /sys/block/$DISC/queue/scheduler
    cat /sys/block/$DISC/queue/scheduler
    sync && /sbin/hdparm -tT /dev/$DISC && echo "----"
    sleep 7
done
```

Данный скрипт замеряет скорость работы диска с помощью утилиты `hdparm`.

P.S. Для установки нужного планировщика `sudo modprobe [scheduler]`



```
(root@kali)-[/home/kali/Desktop]
# ./test
none [mq-deadline]
[none] mq-deadline

/dev/sda:
Timing cached reads:   36660 MB in  1.99 seconds = 18465.85 MB/sec
Timing buffered disk reads: 2290 MB in  3.00 seconds = 763.06 MB/sec
-----
none [mq-deadline]

/dev/sda:
Timing cached reads:   42820 MB in  1.99 seconds = 21565.65 MB/sec
Timing buffered disk reads: 2874 MB in  3.00 seconds = 957.73 MB/sec
-----
none mq-deadline [bfq]

/dev/sda:
Timing cached reads:   37970 MB in  1.99 seconds = 19101.58 MB/sec
Timing buffered disk reads: 2536 MB in  3.00 seconds = 844.95 MB/sec
-----
none mq-deadline bfq [kyber]

/dev/sda:
Timing cached reads:   33054 MB in  1.99 seconds = 16648.56 MB/sec
Timing buffered disk reads: 2904 MB in  3.00 seconds = 967.89 MB/sec
-----
```

Рисунок 1 – Пример запуска скрипта

### 1.2 Измерения

(6 CPUs, 12 GB RAM, NVMe SSD M.2)

№	Планировщик	Кэшированная оперативная память	Последовательное чтение
1.	none	18465.85 MB/sec	763.06 MB/sec
2.		19833.88 MB/sec	765.36 MB/sec
3.		19122.98 MB/sec	971.98 MB/sec
4.	mq-deadline	21565.65 MB/sec	957.73 MB/sec
5.		19922.53 MB/sec	877.22 MB/sec
6.		21395.67 MB/sec	857.59 MB/sec
7.	bfq	19101.58 MB/sec	844.95 MB/sec
8.		18449.51 MB/sec	924.47 MB/sec
9.		18025.41 MB/sec	839.39 MB/sec
10.	kyber	16648.56 MB/sec	967.89 MB/sec
11.		19037.20 MB/sec	874.85 MB/sec
12.		19505.17 MB/sec	895.91 MB/sec

Среднее:

Планировщик	none	mq-deadline	bfq	kyber
cached reads, MB/sec	19140,90	20961,28	18525,50	18396,98
buffered disk reads, MB/sec	833,47	897,51	869,60	912,88

На второй машине (1 CPU, 4 GB RAM, Macbook Air M1 SSD):

№	Планировщик	Кэшированная оперативная память	Последовательное чтение
1.	none	15534,1	602,07
2.		14941,4	610,41
3.		15320,4	600,35
4.	mq-deadline	15153,8	590,77
5.		15755	540,04
6.		15439,7	588,44
7.	bfq	15112	580,2
8.		15326,3	609,37
9.		15513,9	602,72
10.	kyber	15061,8	557,43
11.		15753,7	599,11
12.		14674,8	600,72

Среднее:

Планировщик	none	mq-deadline	bfq	kyber
cached reads, MB/sec	15265,29	15449,52	15317,41	15163,44
buffered disk reads, MB/sec	604,28	573,08	597,43	585,75

### 1.3 Анализ результатов

Планировщик mq-deadline показал наилучшую производительность при чтении из кэша (cached reads) на обеих машинах.

Однако при буферизованном чтении (buffered disk reads) результаты варьируются. На первой машине (с NVMe SSD и 6 CPU) kyber показал лучшую скорость, тогда как на второй машине (с 1 CPU и SSD) небольшое преимущество у none и bfq.

На менее мощной системе с одним процессором, где конкуренция за диск ниже, роль планировщика снижается, и результат больше зависит от возможностей самого SSD.

## 2 МОДИФИКАЦИЯ ПЛАНИРОВЩИКА НА УРОВНЕ ЯДРА

### 2.1 Модификация исходного кода ядра linux

Скачиваем код ядра

```
apt-get source linux-image-$(uname -r)
```

Может понадобиться добавить в файл `/etc/apt/sources.list` строки:

```
deb http://http.kali.org/kali kali-rolling main contrib non-free
deb-src http://http.kali.org/kali kali-rolling main contrib non-free
```

Далее внесем изменения в файл `linux-6.10.11/block/kyber-iosched.c`:

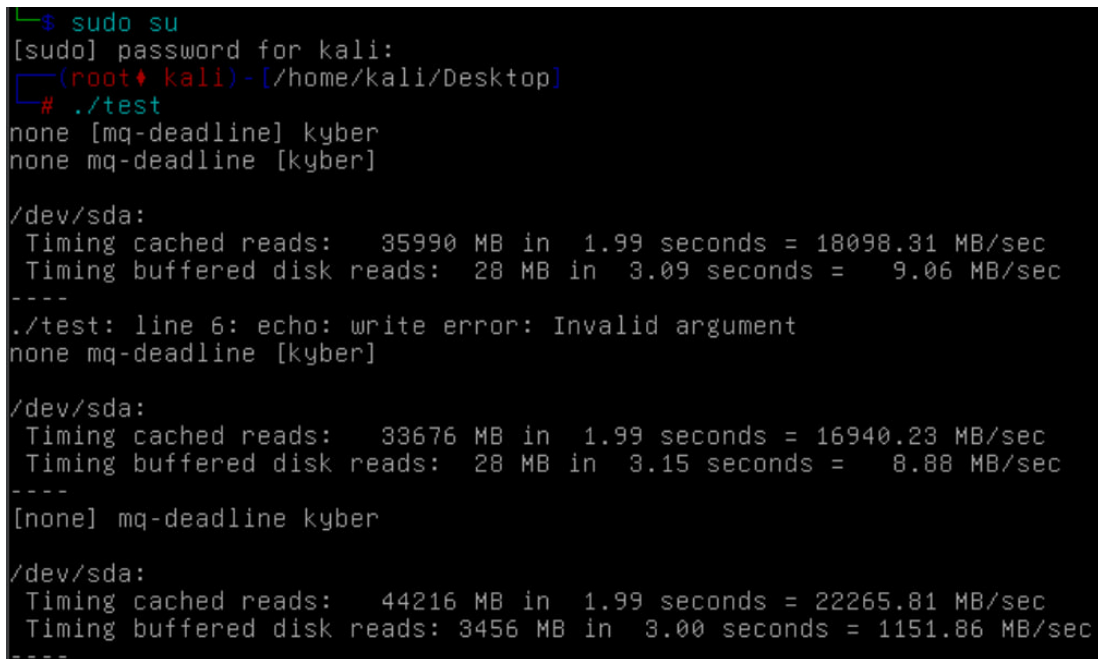
```
#include <linux/delay.h>...
static struct request *kyber_dispatch_request(struct blk_mq_hw_ctx *hctx)
{...
    mdelay(100); // 100 миллисекунд задержки...
```

Соберём ядро:

```
make defconfig
make prepare
make -j$(nproc)
make modules_install
make install
```

После перезагрузки в меню GRUB запустим наше ядро с параметрами в строке, содержащей `linux` добавим в конец `3 single` для запуска в текстовом режиме.

Пробуем теперь запустить наш test



```
$ sudo su
[sudo] password for kali:
(root)kali-[/home/kali/Desktop]
# ./test
none [mq-deadline] kyber
none mq-deadline [kyber]

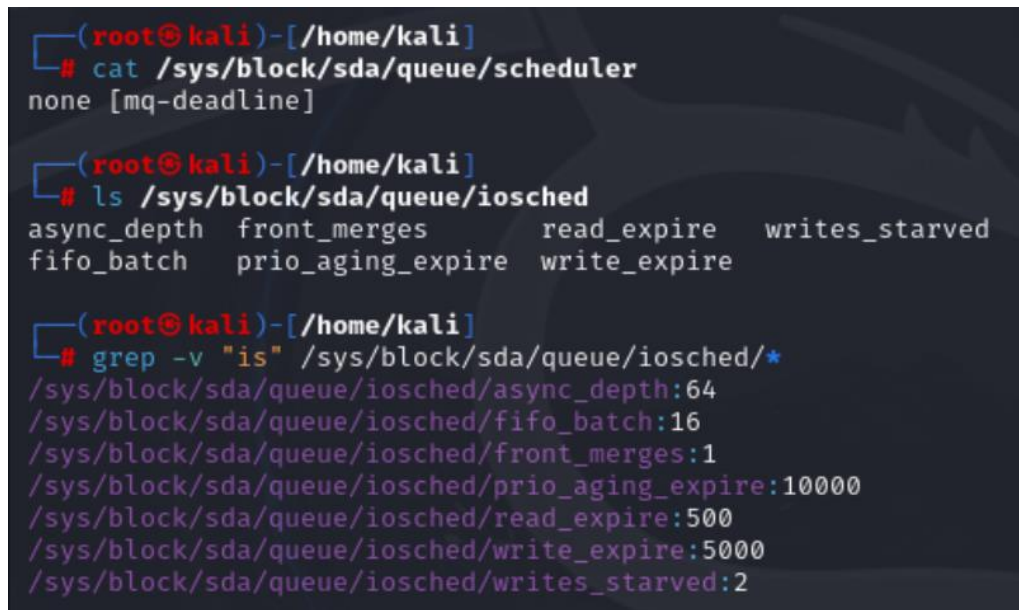
/dev/sda:
Timing cached reads:   35990 MB in  1.99 seconds = 18098.31 MB/sec
Timing buffered disk reads:  28 MB in  3.09 seconds =  9.06 MB/sec
----
./test: line 6: echo: write error: Invalid argument
none mq-deadline [kyber]

/dev/sda:
Timing cached reads:   33676 MB in  1.99 seconds = 16940.23 MB/sec
Timing buffered disk reads:  28 MB in  3.15 seconds =  8.88 MB/sec
----
[none] mq-deadline kyber

/dev/sda:
Timing cached reads:   44216 MB in  1.99 seconds = 22265.81 MB/sec
Timing buffered disk reads: 3456 MB in  3.00 seconds = 1151.86 MB/sec
----
```

Рисунок 2 – Запуск с модифицированным ядром

## 2.2 Модификация существующего планировщика на уровне ядра



```
(root@kali)-[/home/kali]
# cat /sys/block/sda/queue/scheduler
none [mq-deadline]

(root@kali)-[/home/kali]
# ls /sys/block/sda/queue/iosched
async_depth  front_merges  read_expire  writes_starved
fifo_batch   prio_aging_expire  write_expire

(root@kali)-[/home/kali]
# grep -v "is" /sys/block/sda/queue/iosched/*
/sys/block/sda/queue/iosched/async_depth:64
/sys/block/sda/queue/iosched/fifo_batch:16
/sys/block/sda/queue/iosched/front_merges:1
/sys/block/sda/queue/iosched/prio_aging_expire:10000
/sys/block/sda/queue/iosched/read_expire:500
/sys/block/sda/queue/iosched/write_expire:5000
/sys/block/sda/queue/iosched/writes_starved:2
```

Рисунок 3 – Параметры планировщика mq-deadline

**async\_depth:** Максимальная глубина асинхронных операций, которые планировщик будет обрабатывать. (до 512)

**fifo\_batch:** Количество операций, которые могут быть обработаны из FIFO-очереди перед переходом к другим очередям. (до 64)

**front\_merges:** Указывает, должны ли операции I/O объединяться, если они находятся в одной очереди и могут быть обработаны последовательно.

**prio\_aging\_expire:** Время, после которого приоритет операции I/O снижается, если она остается в очереди. (до 100000 мс)

**read\_expire:** Время, через которое запрос на чтение считается просроченным. (до 10000 мс)

**write\_expire:** Время, через которое запрос на запись считается просроченным. (до 10000 мс)

**writes\_starved:** Максимальное количество операций записи, которые могут быть "голодными" (не выполняться) перед тем, как планировщик начнет обрабатывать их. (до 10)



```
none [mq-deadline]

/dev/sda:
Timing cached reads:   52640 MB in  1.99 seconds = 26490.43 MB/sec
Timing buffered disk reads: 2736 MB in  3.01 seconds = 909.09 MB/sec
_____
```

Рисунок 4 – Результаты до

```
none [mq-deadline]

/dev/sda:
Timing cached reads:   52968 MB in  1.99 seconds = 26658.56 MB/sec
Timing buffered disk reads: 2068 MB in  3.01 seconds = 687.92 MB/sec
_____
```

Рисунок 5 – `echo 512 > /sys/block/sda/queue/iosched/async_depth`

```
none [mq-deadline]

/dev/sda:
Timing cached reads:   47192 MB in  1.99 seconds = 23741.12 MB/sec
Timing buffered disk reads: 2292 MB in  3.01 seconds = 762.12 MB/sec
_____
```

Рисунок 6 – `echo 64 > /sys/block/sda/queue/iosched/fifo_batch`

```
none [mq-deadline]

/dev/sda:
Timing cached reads:   43078 MB in  1.99 seconds = 21676.83 MB/sec
Timing buffered disk reads: 2802 MB in  3.00 seconds = 932.98 MB/sec
_____
```

Рисунок 7 – `echo 1 > /sys/block/sda/queue/iosched/prio_aging_expire`

```
none [mq-deadline]

/dev/sda:
Timing cached reads:   56486 MB in  1.99 seconds = 28437.27 MB/sec
Timing buffered disk reads: 2972 MB in  3.00 seconds = 990.38 MB/sec
_____
```

Рисунок 8 – `echo 10000 > /sys/block/sda/queue/iosched/read_expire`

```
none [mq-deadline]

/dev/sda:
Timing cached reads:   44142 MB in  1.99 seconds = 22229.07 MB/sec
Timing buffered disk reads: 3074 MB in  3.00 seconds = 1024.05 MB/sec
_____
```

Рисунок 9 – `echo 10000 > /sys/block/sda/queue/iosched/write_expire`

```
none [mq-deadline]

/dev/sda:
Timing cached reads:   49140 MB in  1.99 seconds = 24722.60 MB/sec
Timing buffered disk reads: 2458 MB in  3.00 seconds = 819.21 MB/sec
_____
```

Рисунок 10 – `echo 10 > /sys/block/sda/queue/iosched/writes_starved`

В теории:

Наиболее значительное ухудшение производительности произойдет при увеличении `read_expire` и `write_expire` до их максимальных значений, так как это приведет к существенному увеличению задержек для операций чтения и записи. Остальные параметры могут сильно повлиять на производительность очень слабой системы.

## **ЗАКЛЮЧЕНИЕ**

В ходе данной лабораторной работы тестирование показало, что выбор планировщика ввода-вывода имеет критическое значение для оптимизации производительности системы. В условиях высокой нагрузки и необходимости минимизации задержек, mq-deadline показывает наилучшие результаты, однако выбор оптимального планировщика зависит от конкретных требований к системе и характера выполняемых задач.

Так же модификация планировщика на уровне ядра продемонстрировала возможность изменения его параметров для более эффективной работы в условиях специфических нагрузок.

Мы выяснили, что рекомендуется проводить периодическое тестирование планировщиков для выбора наилучшего в условиях изменяющихся требований к производительности.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. [Linux I/O Scheduler. Выбираем оптимальный / Хабр](#)
2. [Сравниваем планировщики ввода/вывода ядра Linux — Хакер](#)
3. [blk-mq и планировщики ввода-вывода / Хабр](#)
4. [Планировщик ввода / вывода BFQ лучше / Хабр](#)
5. [OpenNET: статья - Планировщики ввода/вывода и процессов в Linux \(io linux scheduler cfq ionice process\)](#)
6. [I/O Schedulers](#)
7. [kernel.org/doc/Documentation/block/cfq-iosched.txt](#)
8. [kernel.org/doc/Documentation/block/bfq-iosched.txt](#)
9. [Disk scheduling](#)
10. [BFQ \(Budget Fair Queueing\) — The Linux Kernel documentation](#)