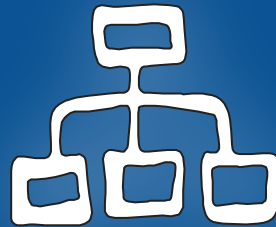


# Спец. семинар «Корректность программ и операционные системы»

Семинар #1:

1. Структура курса, цели и задачи.
2. Организация многопоточности в ОС.
3. Прimitives блокирующей синхронизации: мьютекс, семафор.

# Структура курса. Цели и задачи.



# Операционные системы

**Цель:** проектирование и разработка ПО в рамках ОС.

**Задачи:**

- Повторение и закрепление простейших API ОС Linux.
- Рассмотрение альтернатив и сравнение подходов.

**В рамках ОС Linux:**

- Организация многопоточных вычислений.
- Блокирующая/неблокирующая синхронизация потоков.
- Простейший lock-free алгоритм.
- Синхронные и асинхронные интерфейсы ввода-вывода.
- Мультиплексирование потоков данных.
- Организация сетевого взаимодействия: модели UDP/IP, TCP/IP.

# Корректность программ

**Цель:** формирование базовых прикладных навыков верификации.

**Задачи:**

- Знакомство с необходимыми инструментами.
- Выращивание собственной инфраструктуры верификации.

**На примере одного проекта:**

- Работа с нетривиальными критериями корректности.
- Разработка тестовой системы и тестовых сценариев.
- Применение статического анализатора.
- Разработка проектной документации.

# Система оценивания и план коммуникации

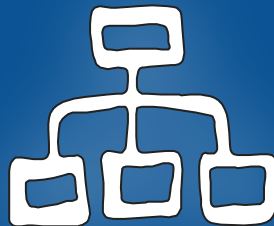
## Система оценивания:

1. Недифференцированный зачёт.
2. Есть один проект. В рамках проекта 10 требований.
3. Условие зачёта – выполнение всех 10 требований.

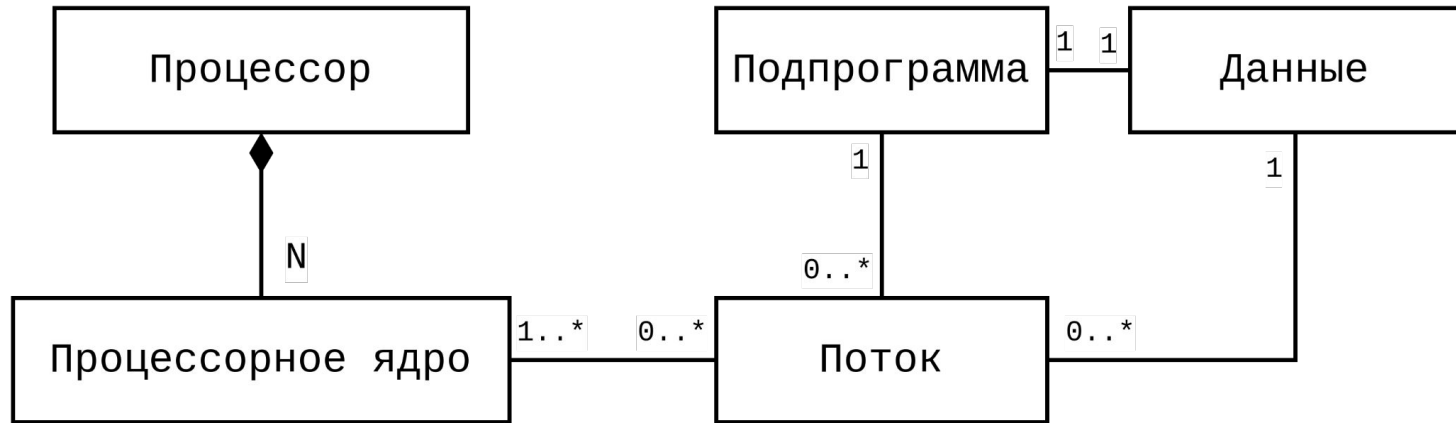
## План коммуникации:

1. Источник информации – чат “Семинар 2024/3 курс”.
2. 1-ый семинар – 24 февраля.
3. 2-ой семинар – 3 марта.
4. Далее – семинары каждую вторую неделю.
5. Сдача: после семинара или в другой день по договорённости.

# Организация многопоточности в операционной системе



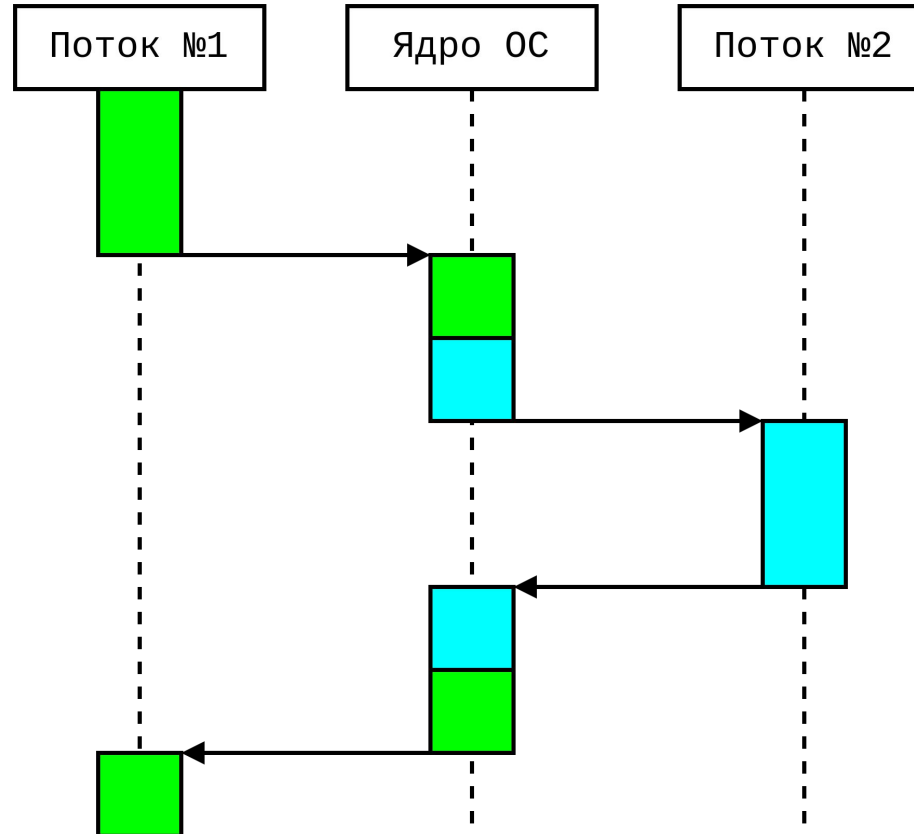
# Модель многопоточности



## Технические особенности реализации модели:

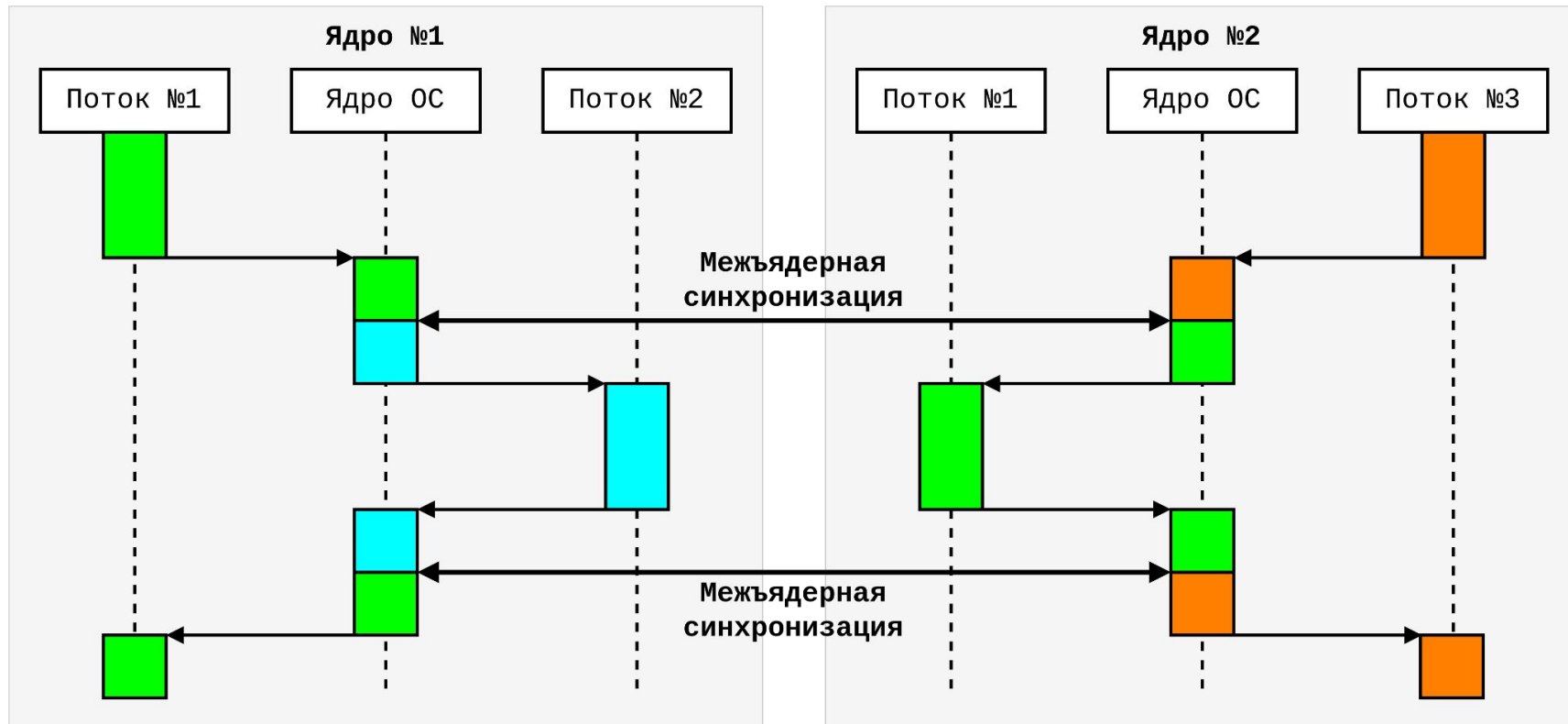
- Переключение и миграция потоков, аффинность потока.
- Разделение ресурсов процессора между ядрами.
- Разделение данных между потоками, гонки по данным.
- Идея Thread Local Storage.

# Переключение потоков на одном ядре

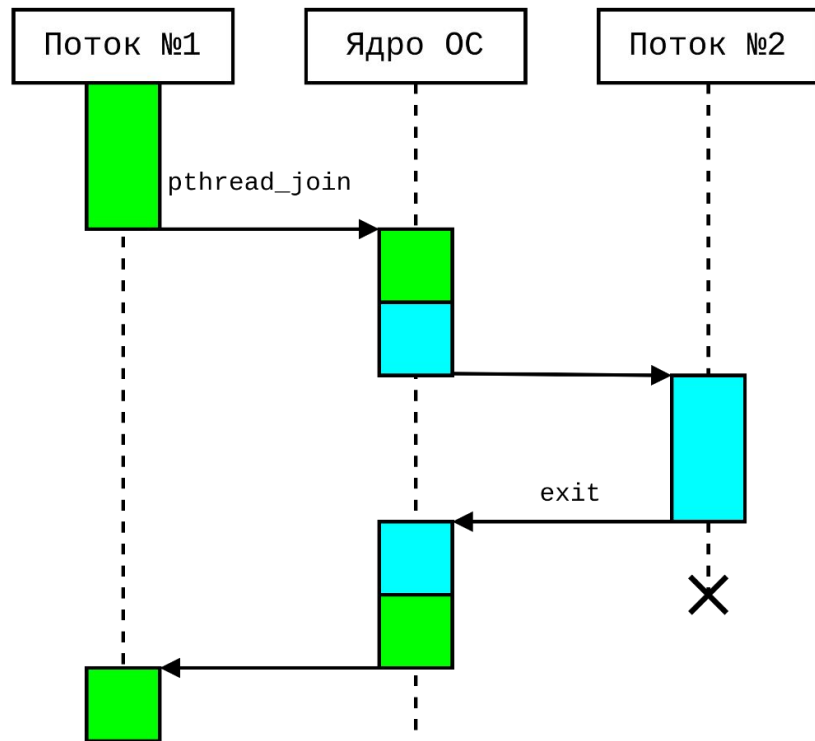
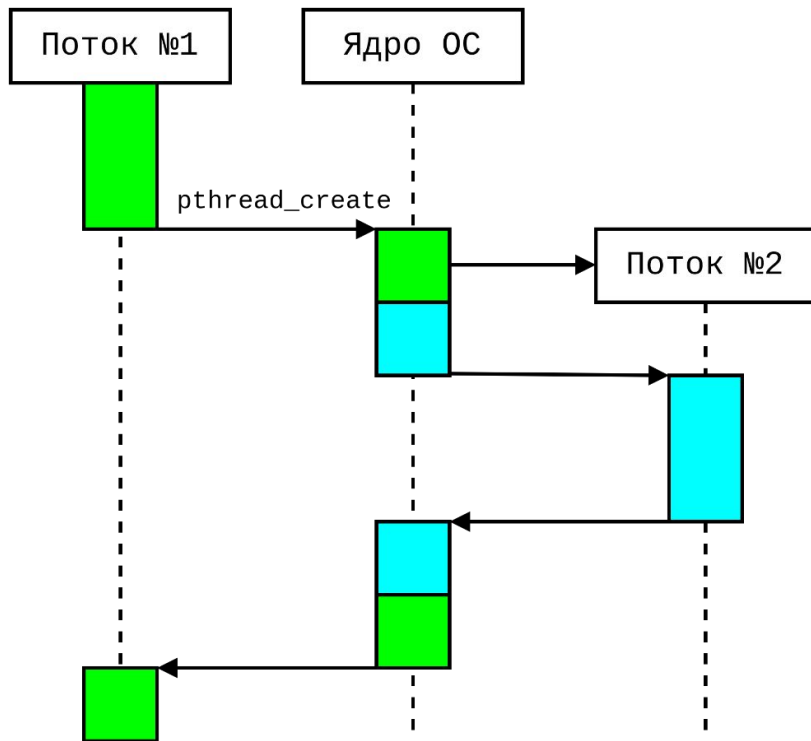




# Миграция потоков между ядрами



# Многопоточность при помощи POSIX Threads



# Простейшая гонка по данным

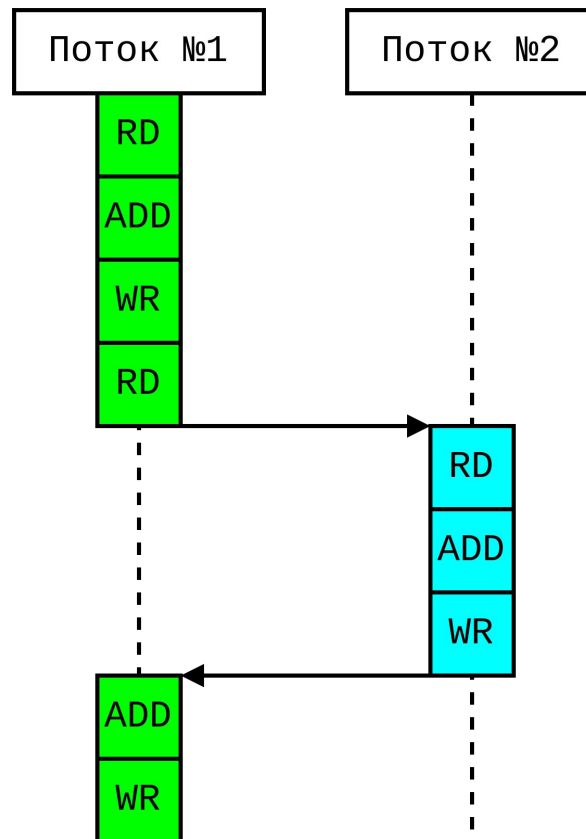
```
mov    eax, dword [rel var]
add    eax, 0x1
mov    dword [rel var], eax
add    qword [rbp-0x10 {i}], 0x1

mov    eax, 0x989680
cmp    qword [rbp-0x10 {i}], rax
jb     0x1289
```

**sequenced-before( $T_i$ )** – отношения порядка для побочных эффектов выражений в C в рамках одного потока.

**happens-before** – отношение порядка для побочных эффектов в рамках нескольких потоков.

**sequenced-before( $T_i$ )  $\subseteq$  happens-before**



# Примитивы блокирующей синхронизации



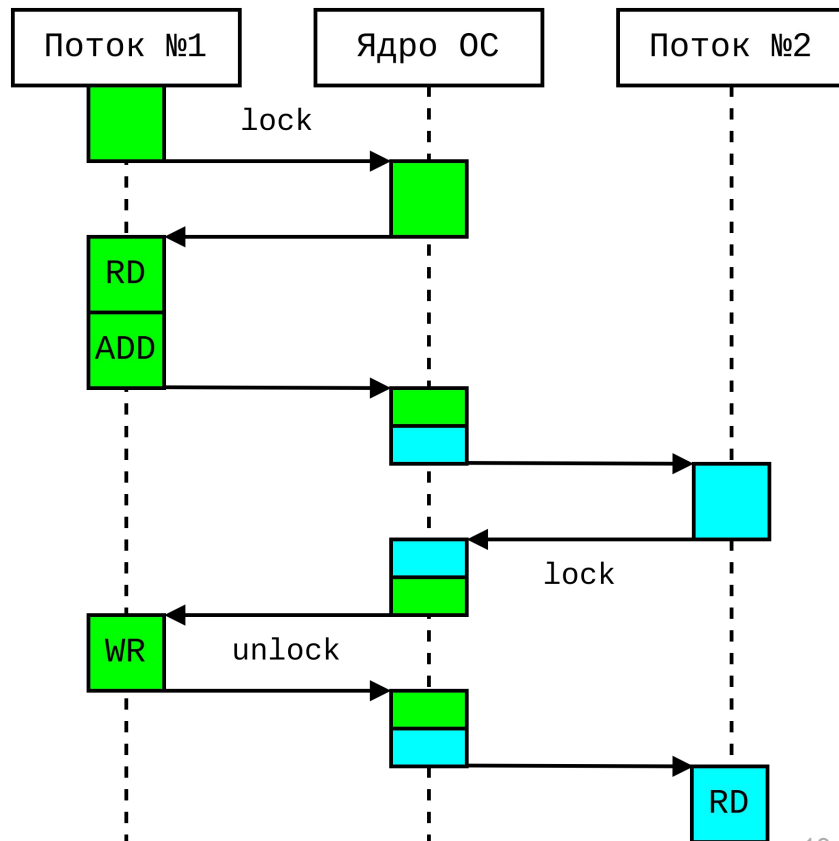
# Примитивы блокирующей синхронизации

## Задача примитива синхронизации:

- Создание критической секции.
- Создание отношения `inter-thread-happens-before`.
- Разделённый доступ к ресурсу.

## Разные примитивы синхронизации:

- Мьютекс: бит + очередь.
- POXIX семафор: число + очередь.
- System-V семафор: N чисел + очередь.



**Спасибо за внимание!**  
**Вопросы?**

