Читатели и писатели

Читатели и писатели

Lock Striping в хэш-таблице:

Потоки, которые выполняют поиск в пределах одного страйпа, могут работать параллельно, поскольку они не изменяют состояния хэш-таблицы.

Потоки, которые вставляют и удаляют элементы, требуют взаимного исключения в пределах страйпа.

Потоки, которые выполняют поиск, назовем **читателями**, а потоки, которые запускают вставки и удаления – **писателями**.

Роли могут меняться: поток, который выполнил вставку элемента, а затем начал поиск, превращается из писателя в читателя.

Readers–Writers Problem

Пусть с разделяемыми данными работают несколько потоков.

Потоки, которые хотят изменять эти данные, назовем **писателями** (*writers*), а потоки, которые хотят только читать эти данные – **читателями** (*readers*).

Задача:

Придумать механизм синхронизации, который гарантирует эксклюзивный доступ для писателей и позволяет читателям в отсутствии писателей работать конкурентно.

Readers–Writer Mutex

Операции:

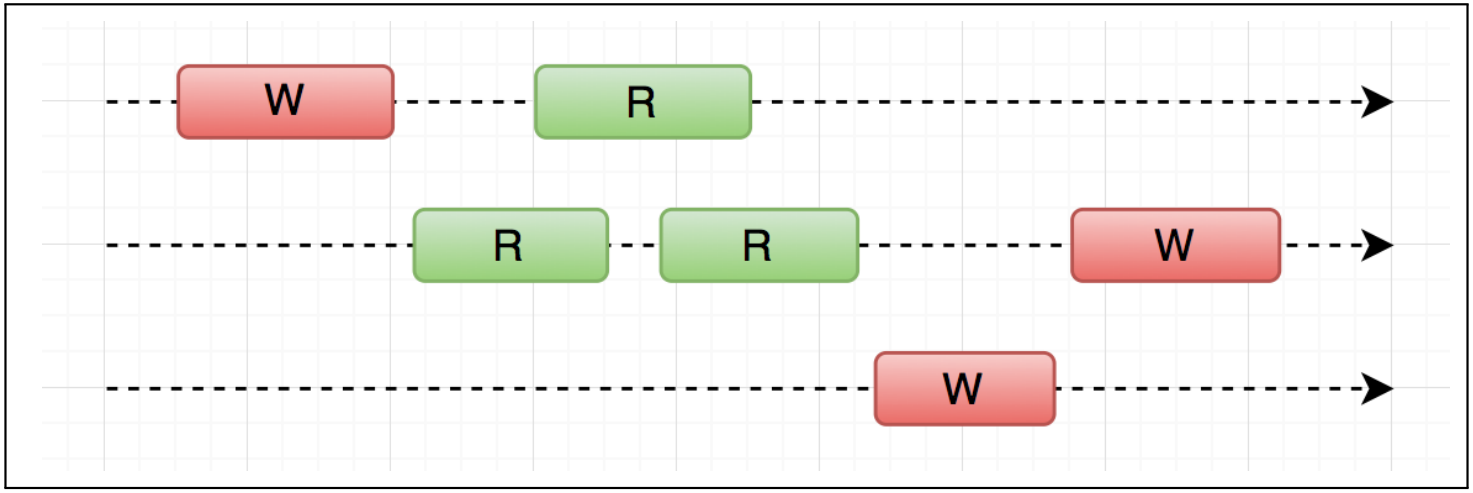
```
rw_mutex.write_lock()      rw_mutex.read_lock()
// critical section        // reader section
rw_mutex.write_unlock()    rw_mutex.read_unlock()
```

Свойства:

- Внутри секции чтения могут одновременно находиться несколько читателей.
- Внутри критической секции может находиться только один писатель.
- Если писатель находится в критической секции, то ни один не может находиться в секции чтения.

Читатели

Секции читателей могут накладываться друг на друга:



Секции писателей не пересекаются с другими.

Readers–Writer Mutex

Состояние `RW-mutex` описывается флагом `writing` и счетчиком `readers`:

- `writing == true` – писатель находится в критической секции
- `readers > 0` – несколько читателей находятся в секции чтения

Два этих состояния взаимоисключающие:

> **Инвариант**: либо `writing == true`, либо `readers > 0`

Свободен: `writing = false, readers = 0`

Реализация на условных переменных

Будем явно хранить состояние `RW-mutex`: флаг `writing` и счетчик `readers`.

Предикаты:

- Читатель не может войти в секцию чтения, пока видит `writing == true`.
- Писатель не может войти в критическую секцию, пока `writing == true` или `readers > 0`.

Условные переменные

Условные переменные (*condition variables*) – механизм блокирующего ожидания и сигнализирования об изменении состояния разделяемого объекта.

Ожидание	Сигнализирование
`cv.wait(mtx)`	`cv.signal()` `cv.broadcast()`

- `cv.wait(mtx)` – заснуть на ожидании сигнала
- `cv.signal()` – послать сигнал одному спящему потоку
- `cv.broadcast()` – разбудить все спящие потоки

Вход в секцию чтения/записи

Инвариант: либо `writing == true`, либо `readers > 0`

Писатель (`write_lock`)

```
gate.lock()
while (writing || readers > 0):
    room_empty.wait(gate)
writing = true
gate.unlock()
```

Читатель (`read_lock`)

```
gate.lock()
while (writing):
    room_empty.wait(gate)
readers += 1
gate.unlock()
```

Блокировка при захвате происходит на условных переменных.

Мьютекс gate захватывается потоками только на входе и на выходе, внутри секций потоки не держат его.

Выход из секции

Писатель (`write_unlock`)

```
gate.lock()
writing = false
room_empty.broadcast()
gate.unlock()
```

Читатель (`read_unlock`)

```
gate.lock()
readers -= 1
if readers == 0:
    room_empty.signal()
gate.unlock()
```

Писатель должен разбудить все ждущие потоки: его могли ждать несколько читателей, все они могут пройти в секцию чтения.

Читателю достаточно разбудить только один поток: его могли ждать только писатели.

Голодание писателей

Писатель (`write_lock`)

```
gate.lock()
while (writing || readers > 0):
    room_empty.wait(gate)
writing = true
gate.unlock()
```

Путь `RW-mutex` захвачен читателями.

Если читатели приходят непрерывно и счетчик `readers` не опускается до нуля, то писатель никогда не сможет войти в критическую секцию.

Приоритет у читателей!

Writer-Priority RW-Mutex

Писатель (`write_lock`)

```
gate.lock()
writers += 1
while (writing || readers > 0):
    room_empty.wait(gate)
writing = true
gate.unlock()
```

Читатель (`read_lock`)

```
gate.lock()
while (writers > 0):
    room_empty.wait(gate)
readers += 1
gate.unlock()
```

- `writing` – писатель захватил эксклюзивный доступ
- `writers` – число писателей (с учетом захватившего)

Writer-Priority RW-Mutex

Первый писатель блокирует новых читателей:

После того, как писатель увеличил счетчик `writers`, новые читатели не смогут проходить в секцию чтения, даже если `RW-mutex` находится в режиме читателей.

Читатели, которые уже выполняют свои секции чтения, завершат их и пропустят писателя.

Выход из секции

В `read_unlock()` теперь нужен `room_empty.broadcast()` вместо `room_empty.signal()`.

Когда последний читатель выходит из секции, то теперь на `room_empty.wait(gate)` могут ждать не только писатели, но и читатели, которые заблокированы ждущими писателями.

Если вызвать `room_empty.signal()`, то может быть получен читателем, который не сможет пройти в критическую секцию из-за `writers > 0`.

Чтобы гарантированно достучаться до писателей, нужно отправить сигнал всем потокам.

Writer-Priority RW-Mutex

Голодание читателей:

Если писатели приходят непрерывно, то счетчик `writers` держится выше нуля и не дает проходить новым читателям.

Приоритет теперь у писателей!

Честный RW-Mutex

Писатель

write_lock()

```
turnstile.lock()
access.wait()
turnstile.unlock()
```

write_unlock()

```
access.signal()
```

Читатель

read_lock()

```
turnstile.lock()
turnstile.unlock()
lightswitch.lock()
readers += 1
if (readers == 1):
    access.wait()
lightswitch.unlock()
```

read_unlock()

```
lightswitch.lock()
readers -= 1
if (readers == 0):
    access.signal()
lightswitch.unlock()
```

Первый взгляд

Используем два мьютекса `turnstile` и `lightswitch` и один бинарный семафор `access`.

Access

Критическая секция / секция читателей начинается с захвата семафора `access` и заканчивается с его освобождением (возвращением жетона).

Потоки захватывают `access` в `read_lock` / `write_lock` и освобождают в `read_unlock` / `write_unlock`.

Lightswitch

Пусть есть темная комната с лампой. Посетитель, который входит в комнату первым, включает лампу, а посетитель, который выходит последним, – выключает.

Читатель, который пришел первым, забирает жетон из семафора `access` и тем самым переводит `RW-mutex` в режим читателей.

Читатель, который ушел последним, возвращает жетон в семафор `access`.

Читатели упорядочиваются с помощью мьютекса `lightswitch` и счетчика `readers`.

Блокировка читателей

Первый читатель, прошедший через турникет `turnstile`, блокируется на `access.wait()`, оставляя заблокированным `lightswitch`.

Последующие читатели, прошедшие через `turnstile`, блокируются на `lightswitch.lock()`, пока первый читатель не захватит `access` и не освободит `lightswitch`.

Если читатель захватил `lightswitch` и видит `readers > 0`, то значит `RW-mutex` уже захвачен читателями, и можно проходить дальше, `access` уже захвачен.

Турникет

Проход через турникет `turnstile.lock()` – это барьер для читателей и писателей:

Если поток прошел через `turnstile.lock()`, то он получает право захватить семафор `access`.

Турникет сделаем честным.

С помощью турникета можно блокировать другие потоки!

Борьба с голоданием писателей

Пусть `RW-mutex` находится в режиме читателей, т.е. жетон из `access` находится у читателей.

Тогда первый писатель, прошедший через турникет `turnstile`, заблокируется на `access.wait()`:

Тем самым он заблокирует турникет `turnstile`:

> Через `turnstile.lock()` не смогут проходить новые читатели и писатели.

Читатели, прошедшие через турникет раньше, завершат свои секции, и последний из читателей на выходе вернет жетон в `access`.

Блокировки читателя

Читатель может заблокироваться на:

- `turnstile.lock()` – если писатель заблокировал турникет и не дает проходить новым читателям.
- `lightswitch.lock()` – если поток ждет первого читателя, который пытается захватить `access`.
- `access.wait()` – первый читатель заблокирован писателем, который находится в критической секции.

Семафор

Почему для `access` используем семафор, а не мьютекс?

Потому что `access` захватывает один читатель, а освобождает в общем случае другой!

У мьютекса есть семантика владения: освобождать мьютекс может только тот поток, который его захватил.

Три стратегии

- Приоритет у читателей
- Приоритет у писателей
- Честный `RW-mutex`