badcasedaily1
22 июл в 09:02

Бинарные семафоры на futex через parking_lot_core

Простой 4 мин 571

Блог компании OTUS, Rust*, Системное программирование*

Обзор

Привет, Хабр!

Сегодня рассмотрим, как реализовать собственный бинарный семафор на основе futex и библиотеки `parking_lot_core`.

Зачем семафор, если есть Mutex?

Ошибка, которую часто совершают начинающие разработчики: отождествляют семафор с мьютексом. Это принципиально разные сущности. Мьютекс ассоциирован с конкретным владельцем: поток, захвативший замок, обязан его освободить. У бинарного семафора понятие владельца отсутствует: это просто счётчик в состоянии `0` или `1`, который обнуляется первым, кто успеет. Его дефолтный сценарий — дозирование доступа к внешним или дорогим ресурсам, буферам фиксированного размера, синхронизация продюсер/консюмер при `buffer size = 1`.

Так же сам по себе семафор не требует связки `Mutex + Condvar`, что экономит как на коде, так и на рантайм-переключениях.

Futex — мостик между user space и ядром

Ключ к высокой эффективности семафоров в Linux — системный вызов `futex`. В простейшем случае он вообще не заходит в ядро: атомарные операции и спиннинг происходят целиком в user space. Лишь при контеншенах задействуется переход в kernel mode.

Под семафор нам достаточно двух операций:

- `FUTEX_WAIT` — поток засыпает, если условие не выполнено.
- `FUTEX_WAKE` — пробуждение ровно одного ожидающего.

Эти две примитивные операции — эквивалент POSIX-терминов `down` и `up`.

Прямой вызов `sys_futex` в Rust возможен, но крайне неудобен: потребуется работать через `libc::syscall`, следить за таймаутами, ручками кодировать параметры и быть готовым к undefined behavior на каждом углу. И тут включается наш `parking_lot_core`.

Parking_lot_core

`parking_lot_core` — низкоуровневая библиотека, извлекающая логику парковки из более высокоуровневых примитивов `parking_lot`. Автор библиотеки вдохновился архитектурой WebKit, где управление потоками организовано через абстрактные очереди спящих по адресу.

Основные идеи:

- каждое «состояние» — это ключ в очереди потоков;
- `park`, `unpark_one`, `unpark_all` — API, которое инкапсулирует системные вызовы и MCS-замки.

Так можно строить собственные синхронизационные структуры без необходимости напрямую взаимодействовать с ядром.

Реализация бинарного семафора

Память

Бинарный семафор — это всего один `AtomicU32`.

```
use core::sync::atomic::{AtomicU32, Ordering};
use parking_lot_core::{park, unpark_one, SpinWait, DEFAULT_PARK_TOKEN, DEFAULT_

const AVAILABLE: u32 = 1;
const TAKEN: u32 = 0;
```

```
pub struct BinarySemaphore {
    state: AtomicU32,
}
```

Захват

Захват жетона идёт по принципу «быстрый путь / медленный путь»:

```
pub fn acquire(&self) {
    let mut spin = SpinWait::new();
    loop {
        if self.state
            .compare_exchange(AVAILABLE, TAKEN, Ordering::Acquire, Ordering::Re
            .is_ok()
        {
            return;
        }

        if spin.spin() {
            continue;
        }

        unsafe {
            park(
                &self.state as *const _ as usize,
                || self.state.load(Ordering::Relaxed) == TAKEN,
                || {},
                |_, _| {},
                DEFAULT_PARK_TOKEN,
                None,
            );
        }
        spin.reset();
    }
}
```

Сначала идёт спинлок, чтобы не загружать ядро понапрасну. Если не удаётся — поток паркуется.

Освобождение

```
pub fn release(&self) {
    if self.state.swap(AVAILABLE, Ordering::Release) == TAKEN {
        unsafe {
            unpark_one(
                &self.state as *const _ as usize,
                |_| DEFAULT_UNPARK_TOKEN,
            );
        }
    }
}
```

`unpark_one` будит ровно одного спящего потока.

RAII guard

Чтобы избежать ошибок управления ресурсом, можно обернуть захват в RAII:

```
pub struct SemaphoreGuard<'a>(&'a BinarySemaphore);

impl<'a> Drop for SemaphoreGuard<'a> {
    fn drop(&mut self) {
        self.0.release();
    }
}

impl BinarySemaphore {
    pub fn lock(&self) -> SemaphoreGuard<'_> {
        self.acquire();
        SemaphoreGuard(self)
    }
}
```

Нюансы

Память и порядок операций.
`Acquire` на `compare_exchange` гарантирует, что изменения до `release` видны захватившему. `Release` на `swap` экспонирует все данные до отдачи жетона.

Контекстные переключения.
При высоких нагрузках два потока могут бесконечно «пинг-понговать» жетон, создавая бурю `context switch`. Решение — увеличивать полезную работу в критической секции или переходить на счётный семафор.

Приоритеты.
`futex` не учитывает приоритеты. Для realtime-сценариев лучше использовать `FUTEX_PI` или RT-mutex.

Спуровые пробуждения.
`futex` может разбудить поток без `WAKE`. Именно поэтому после `park` всегда пересчитывается условие.

Переносимость.
Для Windows и macOS есть аналог — крейт `atomic-wait`. Но он не даёт коллбэков под замком, что затрудняет составные конструкции.

Заключение

Бинарный семафор на базе `futex` и `parking_lot_core` — компактный и эффективный примитив: 4 байта памяти, ни одной динамической аллокации и максимум один системный вызов в случае контеншена. Лёгок в реализации и понятен.

Если вам доводилось решать похожие задачи другими способами, или у вас есть соображения, — делитесь опытом и комментариями.

Теги: rust, parking_lot_core, parking_lot, futex, AtomicU32, SpinWait

Хабы: Блог компании OTUS, Rust, Системное программирование

+5 1 0

OTUS

Развиваем технологии, обучая их создателей

Подписаться

Сайт ВКонтакте Telegram

104 Карма 214 Рейтинг

artem @badcasedaily1

Motion – Peter Sandberg

Подписаться

Комментировать

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

the_bat
20 часов назад

Ethernet с дальностью до километра. 10BASE-T1L

Средний 3 мин 9K

+95 61 93

dolgonosic
15 часов назад

Как мы реализовали георезервирование инфраструктуры для системы видеоконференций: опыт, ошибки, выводы

Средний 13 мин 1K

Кейс

+43 30 2

daniilgorbenko
22 часа назад

Ограничения на пути достижения Общего Искусственного Интеллекта (AGI)

Простой 7 мин 5.4K

+42 29 30

alizar
19 часов назад

Математическая живопись Иньиго Килеса

Простой 7 мин 2.5K

+40 42 1

alan_dani
23 часа назад

Паттерны современного Node.js (2025)

Простой 14 мин 5.7K

Перевод

+36 89 8

RationalAnswer
23 часа назад

Кибератаки на Россию, а также ядерный межстрановой щитпостинг

7 мин 7.6K

Дайджест

+34 8 8

Kamil_GR
19 часов назад

Великое вымирание: как ИИ разрушает интернет

Простой 8 мин 6.2K

Мнение

+31 32 88

ifap
18 часов назад

«Вымпелком» – ваше окно в мир спама

Простой 2 мин 4.8K

Кейс

+30 15 19

AndreySy4
20 часов назад

«Щи: симулятор жестокости» или «Как не надо делать игры»

Простой 57 мин 4.4K

Из песочницы

+30 23 9

wwwhttpru
18 часов назад

Опенсорсим yx_state — свой State Management для Flutter

14 мин 1.1K

+21 8 4

35 спикеров, аналитическая дженга и море практики: что будет на конференции для системных и бизнес-аналитиков от Контура

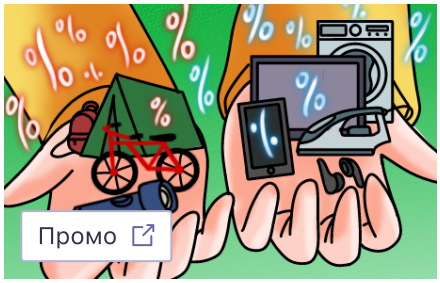
Турбо

Показать еще

МИНУТОЧКУ ВНИМАНИЯ

Делаем сисадмина сильнее с помощью менеджера паролей

Да начнётся битва: выбираем лучший IT-бренд работодателя

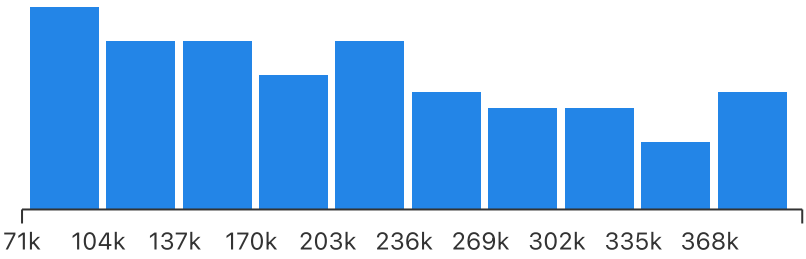


Неважно, что ты выберешь — с Промокодусом будет дешевле

СРЕДНЯЯ ЗАРПЛАТА В IT

213 280 ₽/мес.

— средняя зарплата во всех IT-специализациях по данным из 14 532 анкет, за 2-ое пол. 2025 года. Проверьте «в рынке» ли ваша зарплата или нет!

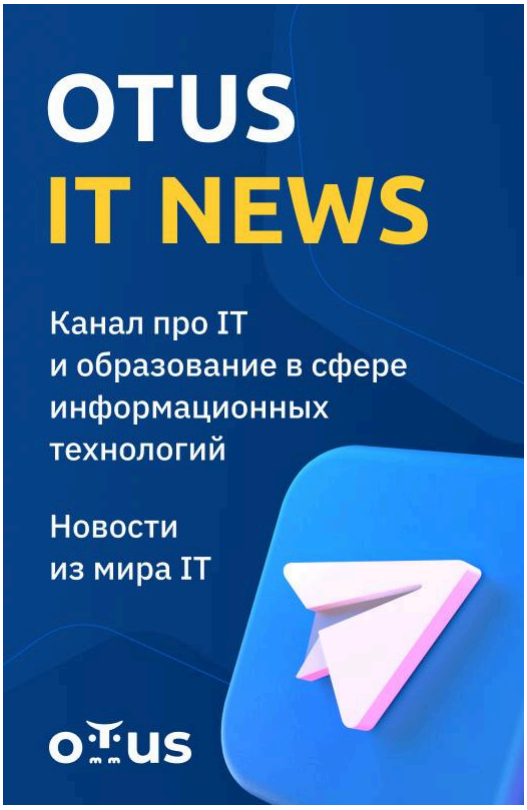


Проверить свою зарплату

ИНФОРМАЦИЯ

Сайт	otus.ru
Дата регистрации	22 марта 2017
Дата основания	1 апреля 2017
Численность	101–200 человек
Местоположение	Россия
Представитель	OTUS

ВИДЖЕТ



23 часа назад

Всё про age в Linux

2.7K 5

2 авг в 13:40

Кейс: Редактирование стандартных документов в Битрикс24

390 1

1 авг в 20:07

Модель 3–3–3: как прогнозировать скорость команды и не попадать в цейтнот

1.9K 1

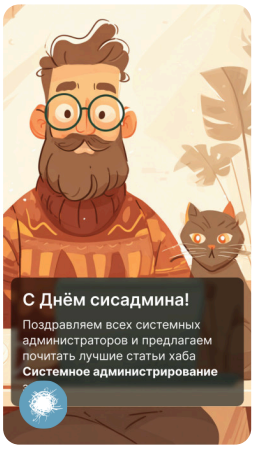
1 авг в 16:25

Всё про std::search и где его применять

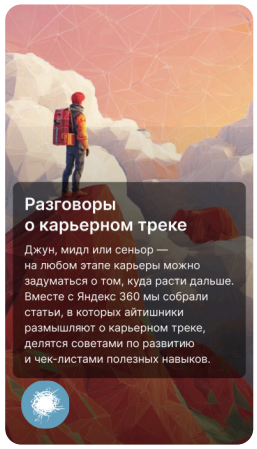
4.2K 6

ИСТОРИИ

История Яндекс Почты



С Днём сисадмина!



Как расти в ИТ: советы, гайды и опыт сеньоров

1 055 450 очков в облаке – попробуй обогнать