

Программирование на современном C++

Обзор многопоточного программирования.



Отметьтесь на портале!

- Посещение необязательное, но тем, кто пришёл, следует отмечаться на портале в начале каждого занятия
- Это позволяет нам анализировать, какие занятия были более или менее интересны студентам, и менять курс в лучшую сторону
- Также это даст возможность вам оставить обратную связь по занятию после его завершения


пн, 1 ноября	вт, 2 ноября	ср, 3 ноября	чт, 4 ноября	пт, 5 ноября	сб, 6 ноября
Нет занятий	<div>18:00 Углубленный C/C++ (w... п</div> <div>Обработка исключительных ситуаций. Шаблоны классов и методов. Обобщенное и безопасное программирование</div> <div>А. Халайджи</div> <div>18:00 Углубленный C/C++ (ML) п</div> <div>Обработка исключительных ситуаций. Шаблоны классов и методов. Обобщенное и безопасное программирование</div> <div>А. Халайджи</div>	Нет занятий	Нет занятий	Нет занятий	Нет занятий

Ссылка на Zoom РК сегодня в 18:00

Безопасность интернет-приложений (третий семестр)

Подключиться к конференции Zoom
<https://mailru.zoom.us/j/98586081818?pwd=eWxwSDdCbHhQNnNwQU5iblFvU2dTZz09>

Идентификатор конференции: 985 8608 1818
Код доступа: 785885

 Алексей Набережный 55 минут назад

★ 0 💬 0 ↓ 0 ↑

Запись прошлой лекции (+ что почитать перед РК)

Безопасность интернет-приложений (третий семестр)

Углублённый C/C++

Лекция 5

📍 **Онлайн - ML**

Отметьтесь, что вы пришли на занятие. Так вы улучшите свою посещаемость и вас увидит преподаватель в своём "Журнале посещений".

Оставьте отзыв о занятии и мы сможем улучшить учебный процесс.

План лекции

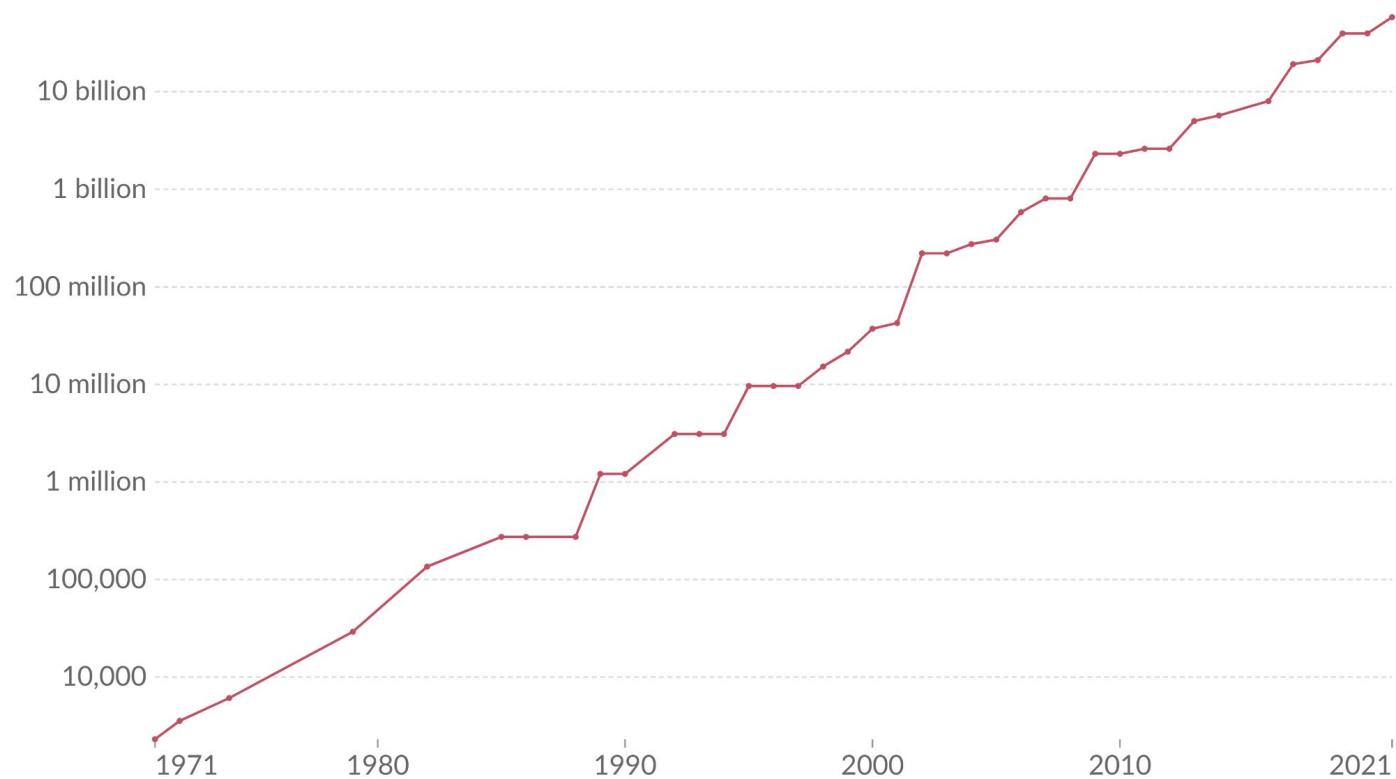
- Возникновение многопоточности
- `std::thread`
- `std::mutex`
- Перерыв
- Практика

Закон Мура (1965)

Moore's law: The number of transistors per microprocessor

The number of transistors that fit into a microprocessor. The observation that the number of transistors on an integrated circuit doubles approximately every two years is called Moore's law¹.

Our World
in Data



Source: Karl Rupp, Microprocessor Trend Data (2022)

OurWorldInData.org/technological-change • CC BY

1. Moore's law: Moore's law is the observation that the number of transistors in a dense integrated circuit doubles about every two years, because of improvements in production. Read more: [What is Moore's Law?](#)

Закон масштабирования Деннарда (1974)

Уменьшая размеры транзистора и повышая тактовую частоту процессора, возможно пропорционально повышать производительность.

Закон масштабирования Деннарда (1974)

Уменьшая размеры транзистора и повышая тактовую частоту процессора, возможно пропорционально повышать производительность.

Техпроцесс приблизился к физическим ограничениям (атом кремния 0,24 нанометра)

Закон масштабирования Деннарда (1974)

Уменьшая размеры транзистора и повышая тактовую частоту процессора, возможно пропорционально повышать производительность.

Техпроцесс приблизился к физическим ограничениям (атом кремния 0,24 нанометра)

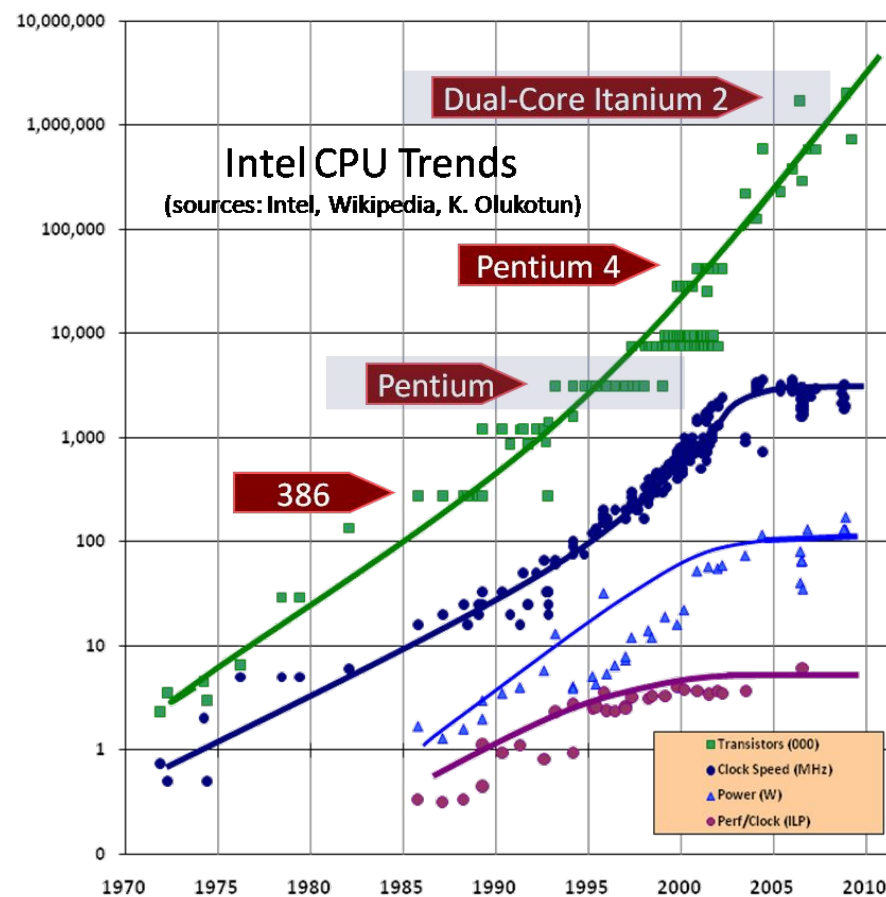
Частоты процессоров ограничиваются возможностью теплоотведения



Герб Саттер

Эксперт по языку C++

The Free Lunch Is Over

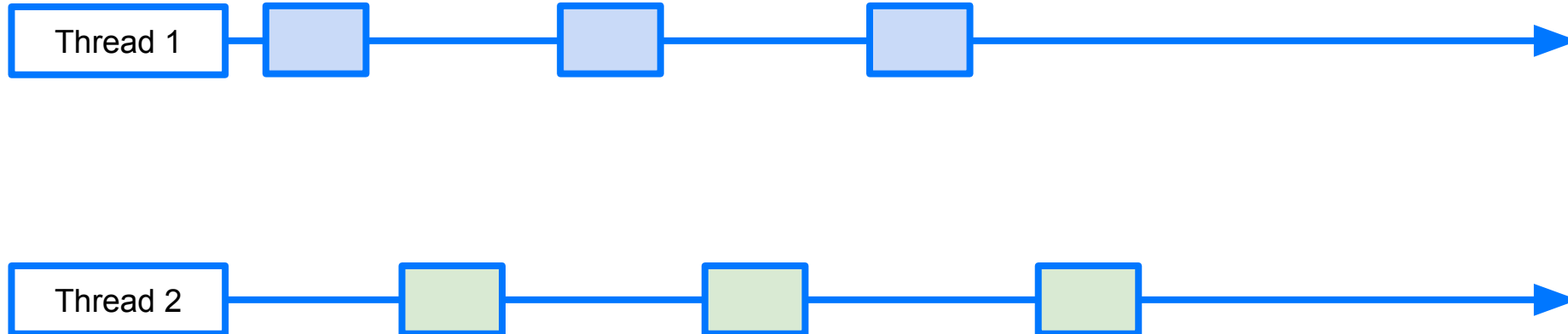


Многопоточность

Свойство платформы или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени.

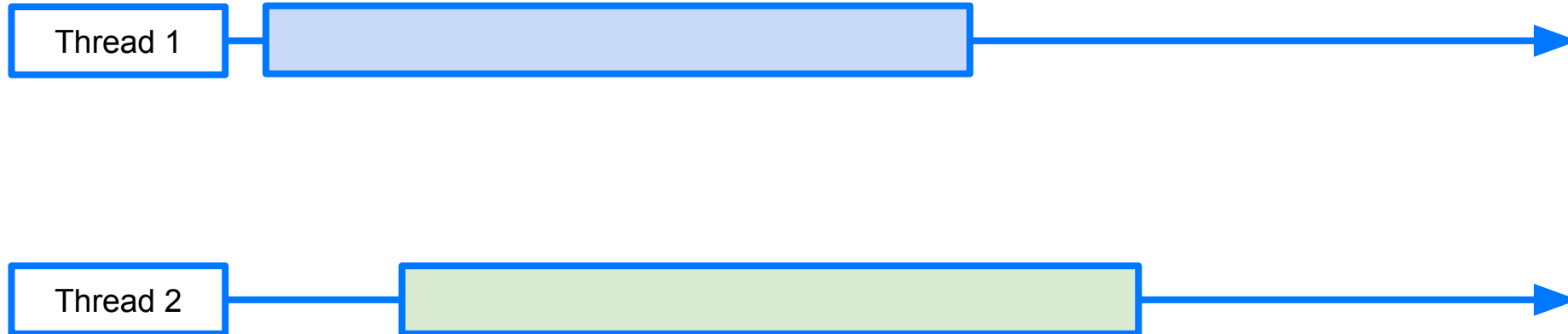
Конкурентность

ВЫПОЛНЕНИЕ НЕ ЯВЛЯЕТСЯ ИСТИННО ПАРАЛЛЕЛЬНЫМ, ЯДРО НЕПРЕРЫВНО ПЕРЕКЛЮЧАЕТСЯ МЕЖДУ ПОТОКАМИ



Параллелизм

ИСТИННАЯ МНОГОПОТОЧНОСТЬ, НУЖЕН МНОГОЯДЕРНЫЙ ПРОЦЕССОР



Concurrency support library

```
int main(int argc, char* argv[]) {  
    std::string str;  
    std::thread thread{[&str]() {  
        str = "Hello from thread";  
    }};  
}
```

В КОНСТРУКТОРЕ - ЛЮБОЙ ВЫЗЫВАЕМЫЙ ОБЪЕКТ,
ПОСЛЕ ВЫЗОДА ИЗ КОНСТРУКТОРА ПОТОК
НАЧИНАЕТ РАБОТУ

```
thread.join(); // Иначе std::terminate! ЖДЕМ, ПОКА ПОТОК ЗАКОНЧИТ ВЫПОЛНЕНИЕ  
std::cout << str << std::endl;  
return 0;  
}
```

ОБЯЗАТЕЛЬНО ДОЖИДАЕМСЯ ПОТОК ПЕРЕД ВЫЗОВОМ ЕГО
ДЕСТРУКТОРА

Пример

Состояние гонки

Если один поток исполнения пишет в область памяти, а другой **читает** или пишет в эту же область.

Состояние гонки

Если один поток исполнения пишет в область памяти, а другой **читает** или пишет в эту же область.

Для разрешения используются примитивы синхронизации

Состояние гонки

Если один поток исполнения пишет в область памяти, а другой **читает** или пишет в эту же область.

Для разрешения используются примитивы синхронизации

Состояние гонки

Если один поток исполнения пишет в область памяти, а другой **читает** или пишет в эту же область.

Для разрешения используются примитивы синхронизации
(ЛИБО ВЫДЕЛЕНИЕ ВРЕМЕННЫХ РЕСУРСОВ КАЖДОМУ ПОТОКУ - НЕ ВСЕГДА ПОЛУЧАЕТСЯ)

- `std::mutex`
- `std::condition_variable`

Пример

Состояние гонки

Если один поток исполнения пишет в область памяти, а другой **читает** или пишет в эту же область.

Для разрешения используются примитивы синхронизации

- `std::mutex`
- `std::condition_variable`

RAII обертки

- `std::lock_guard`
- `std::unique_lock`
- `std::scoped_lock`

Пример

Перерыв

Практика

Спасибо за внимание!

