OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Telegram

Задаем
вопрос в чат

Вопросы вижу в чате,
могу ответить не сразу

Тема вебинара

Разработчик C++ - базовый курс

Модуль 7: Многопоточность в C++

Дополнительные тонкости и детали

Пальчуковский Евгений

Разработчик ПО

Развиваю технологии финансовых услуг с помощью C++

Email: eugene@palchukovsky.com
Telegram: @palchukovsky

Маршрут вебинара

- once_flag + call_once
- thread_local
- Parallel STL
- Заключение
- Ответы на вопросы

once_flag + call_once

- **call_once** используется, когда нужно гарантировать однократный вызов функции в многопоточной среде

- **once_flag** используется как помощник для call_once

Пример использования: однократная инициализация в многопоточной среде

once_flag + call_once

Полезен если

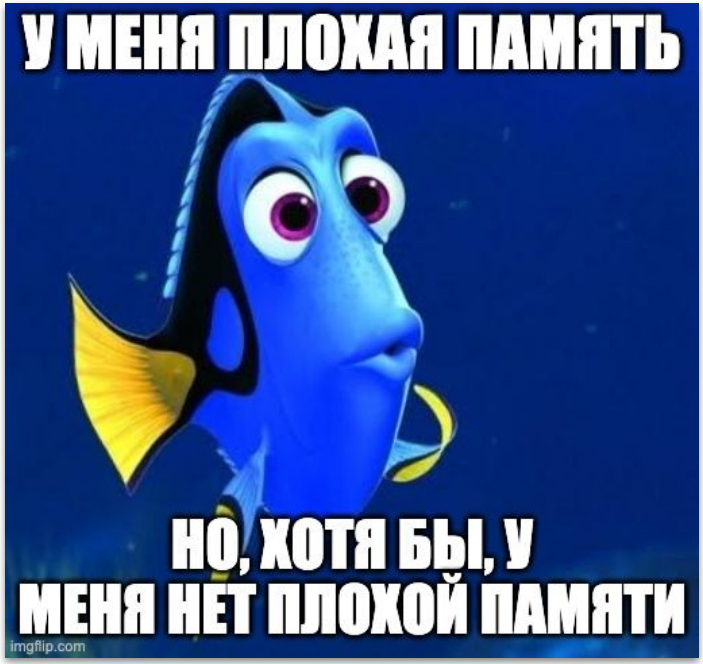
Нужно вызвать **процедуру**
один раз из **нескольких** потоков

+ **невозможно** перенести вызов
  в “**когда еще нет потоков**”

+ нужен именно **вызов процедуры**,
  а не **создание объекта**

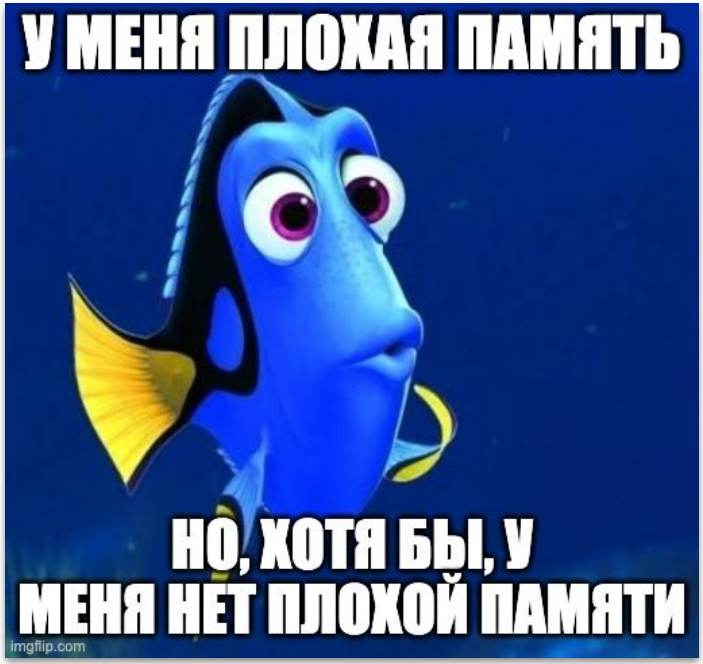
thread_local

- **ключевое слово** для переменной



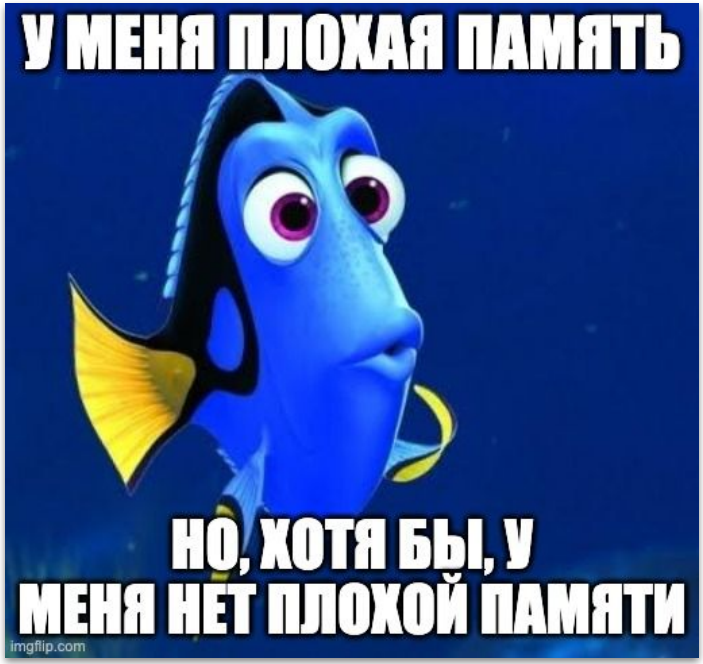
thread_local

- **ключевое слово** для переменной
- каждый поток имеет **свой экземпляр** переменной
- место для переменной выделяется при старте потока
- …и освобождается при его завершении
- переменные **не** нужно защищать мьютексом
- для глобальных и локальных статических



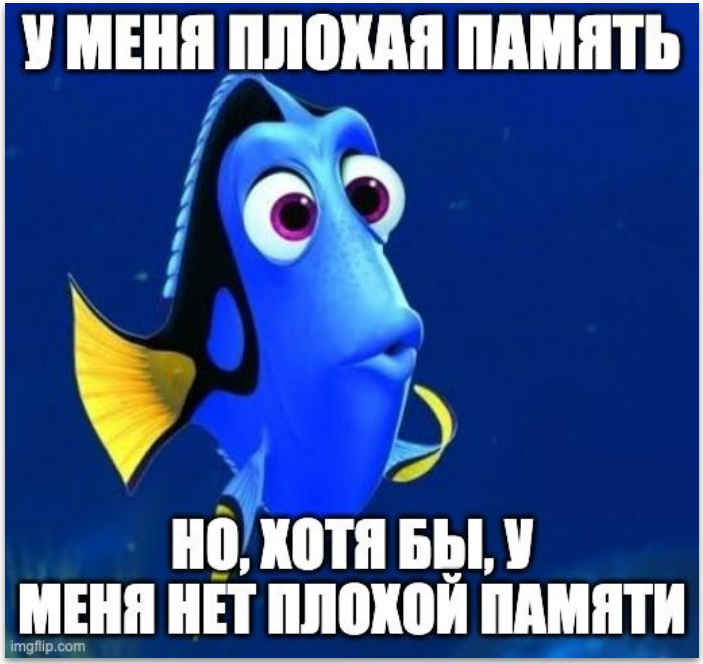
thread_local

- **ключевое слово** для переменной
- каждый поток имеет **свой экземпляр** переменной
- место для переменной выделяется при старте потока
- …и освобождается при его завершении
- переменные не нужно защищать мьютексом
- для глобальных и локальных статических



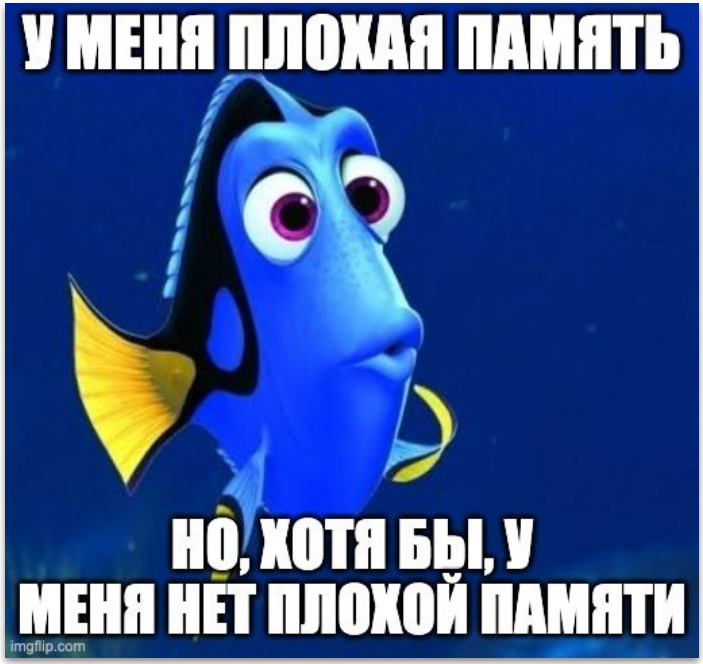
thread_local

- **ключевое слово** для переменной
- каждый поток имеет **свой экземпляр** переменной
- место для переменной выделяется при старте потока
- …и освобождается при его завершении
- переменные **не нужно защищать** мьютексом

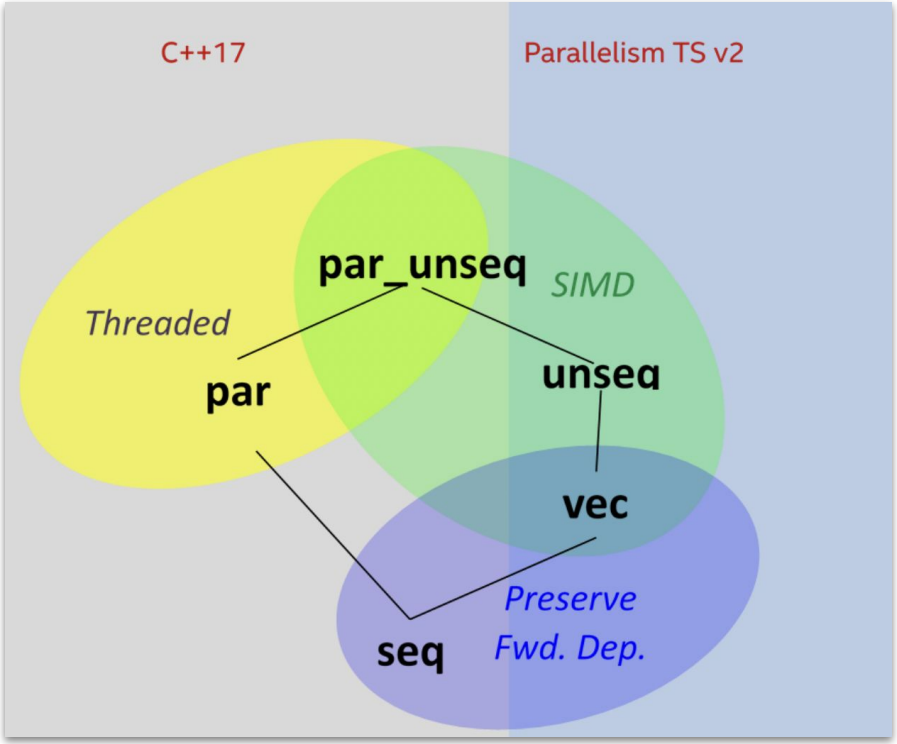


thread_local

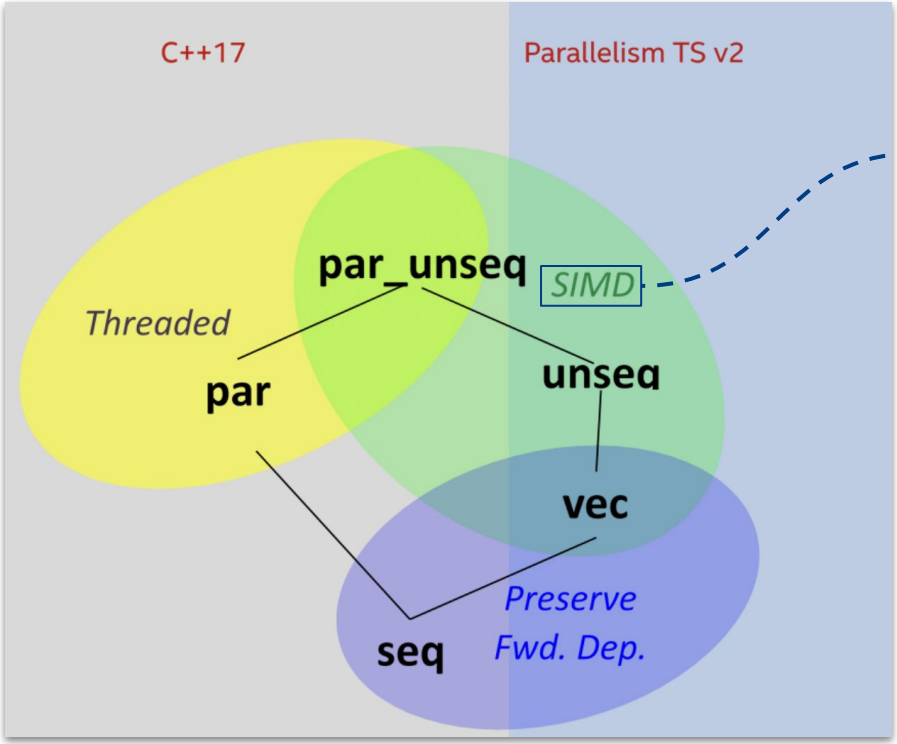
- **ключевое слово** для переменной
- каждый поток имеет **свой экземпляр** переменной
- место для переменной выделяется при старте потока
- …и освобождается при его завершении
- переменные **не нужно защищать** мьютексом
- для **глобальных** и локальных **статических**

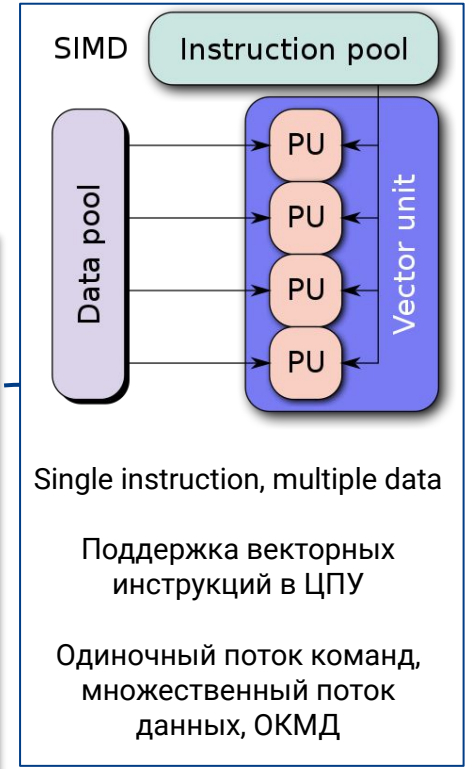


Parallel STL



Parallel STL





Single instruction, multiple data

Поддержка векторных инструкций в ЦПУ

Одиночный поток команд, множественный поток данных, ОКМД

Parallel STL

Многие STL алгоритмы принимают в качестве параметра политику исполнения из пространства **имен std::execution**:

- **seq** - данные обрабатываются в последовательном порядке в один поток
- **par** - данные обрабатываются в несколько потоков, каждый поток обрабатывает данные последовательно
- **unseq** - данные обрабатываются с помощью векторизованных инструкций в один поток
- **par_unseq** - данные обрабатываются в несколько потоков с использованием векторизованных инструкций

Parallel STL

Поддержка в компиляторах:

- **gcc 9** - через библиотеку Intel Threading Building Blocks (Intel TBB)
- **clang 11 (?)** - не требует дополнительных библиотек
- **MSVC 19.14** - не требует дополнительных библиотек

Заключение

1. **once_flag + call_once** - вызвать один раз
   или просто использовать локальную static переменную
2. **thread_local** - каждому потоку по переменной
3. **Parallel STL** - ускоряем многие алгоритмы STL за счет железа

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Заполните, пожалуйста, опрос о занятии по ссылке в чате