



Онлайн образование



Проверить, идет ли запись

Меня хорошо видно && слышно?



Тема вебинара

Понятие потока выполнения программы



Сергей Кольцов

профессиональный программист



Зачем нужны потоки

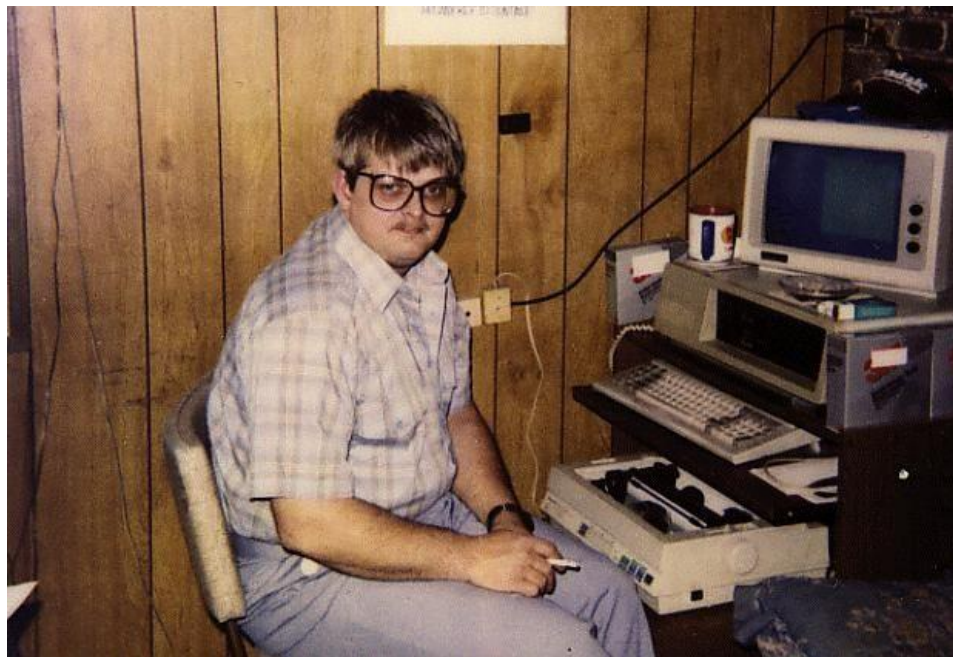
Our World
in Data

This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



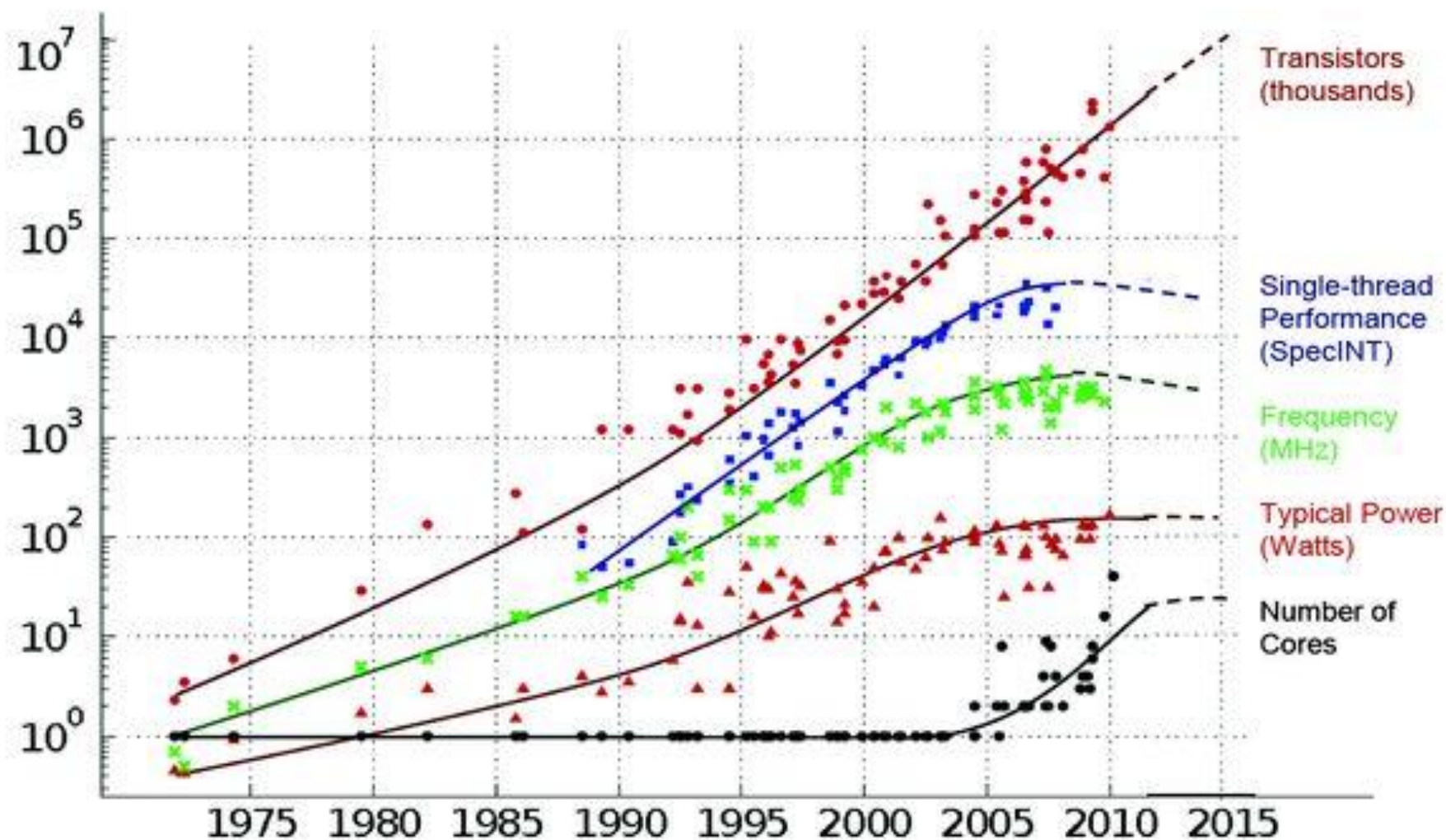
Как менялась тактовая частота?

Какая частота была у вашего первого компьютера?



Напишите в чат частоту в Mhz и год

35 YEARS OF MICROPROCESSOR TREND DATA



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

Потоки и планировщик задач

Многозадачность и многопоточность

Термины:

- **Многозадачность** - свойство ОС одновременно работать над несколькими задачами.



Многозадачность и многопоточность

Термины:

- **Многозадачность** - свойство ОС одновременно работать над несколькими задачами.
- **Многопоточность** - свойство ОС, процесс может иметь несколько потоков.



Многозадачность и многопоточность

Термины:

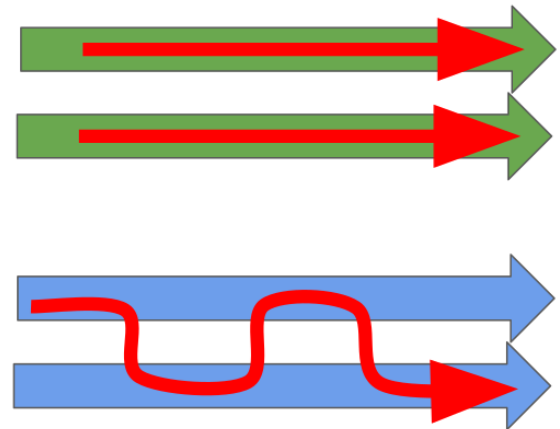
- **Многозадачность** - свойство ОС одновременно работать над несколькими задачами.
- **Многопоточность** - свойство ОС, процесс может иметь несколько потоков.
- **Параллельное выполнение** – одновременная работа вычислительных процессов.



Многозадачность и многопоточность

Термины:

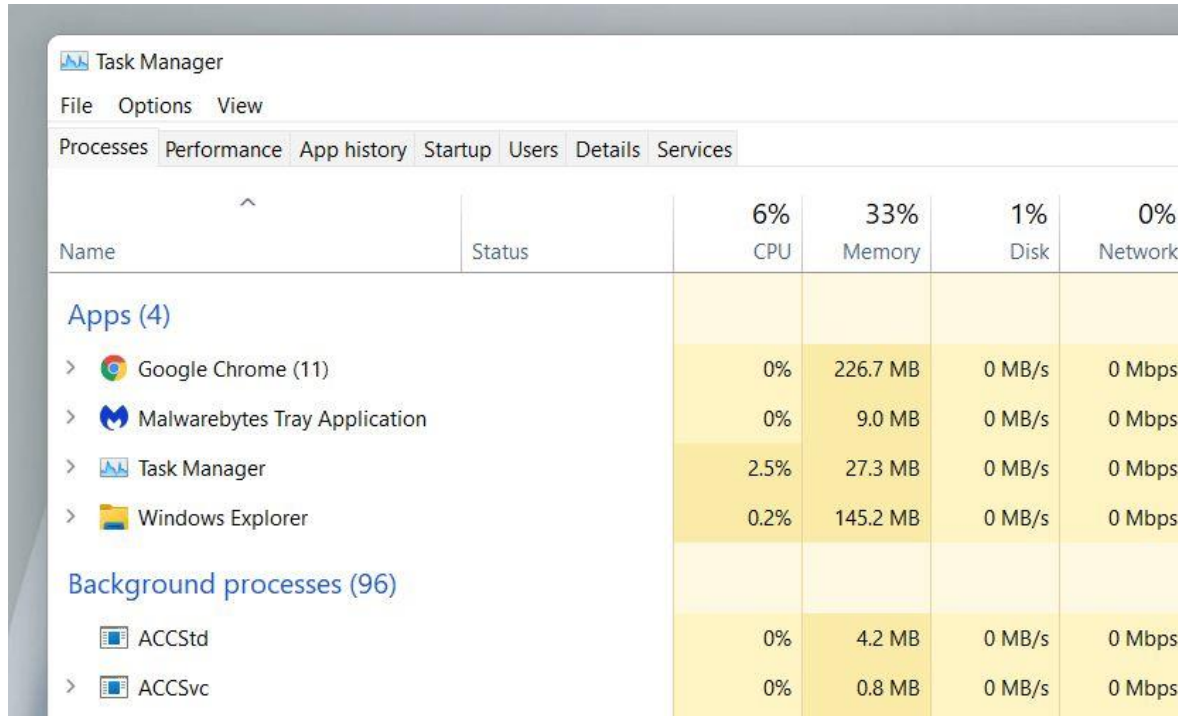
- **Многозадачность** - свойство ОС одновременно работать над несколькими задачами.
- **Многопоточность** - свойство ОС, процесс может иметь несколько потоков.
- **Параллельное выполнение** – одновременная работа вычислительных процессов.
- **Псевдопараллельное выполнение** – последовательная работа вычислительных процессов малыми квантами времени.



Процессы и потоки

Процесс:

- Для пользователя - исполняемый экземпляр программы



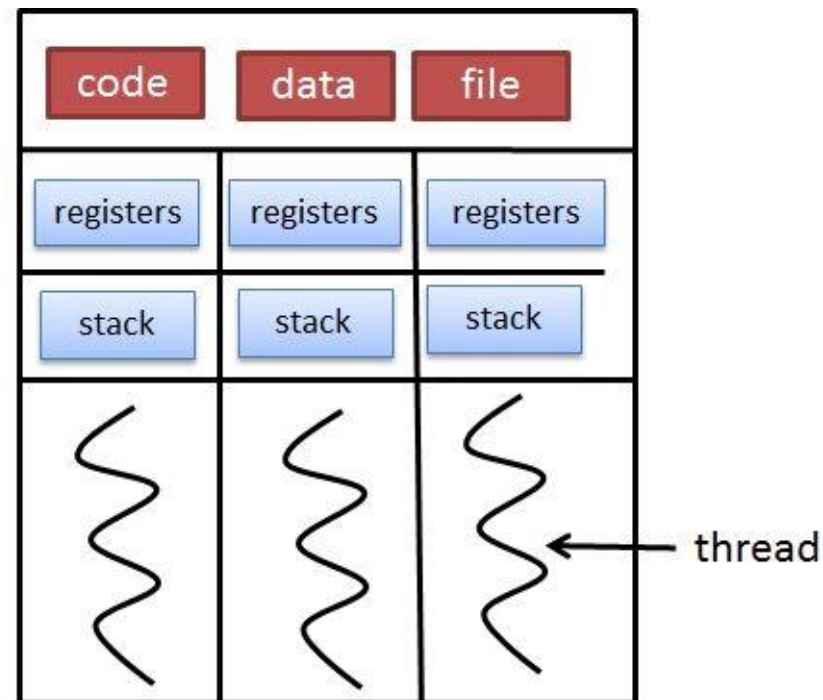
The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. It displays a list of running applications and background processes, categorized into 'Apps (4)' and 'Background processes (96)'. Each process entry includes its name, status, and resource usage (CPU, Memory, Disk, and Network).

Name	Status	6% CPU	33% Memory	1% Disk	0% Network
Apps (4)					
> Google Chrome (11)		0%	226.7 MB	0 MB/s	0 Mbps
> Malwarebytes Tray Application		0%	9.0 MB	0 MB/s	0 Mbps
> Task Manager		2.5%	27.3 MB	0 MB/s	0 Mbps
> Windows Explorer		0.2%	145.2 MB	0 MB/s	0 Mbps
Background processes (96)					
ACCSStd		0%	4.2 MB	0 MB/s	0 Mbps
> ACCSvc		0%	0.8 MB	0 MB/s	0 Mbps

Процессы и потоки

Процесс:

- **Для пользователя** - исполняемый экземпляр программы
- **Для ОС** - контейнер, в котором хранятся ресурсы запущенной программы:
 - адресное пространство
 - потоки
 - открытые файлы
 - дочерние процессы
 - ... и прочее

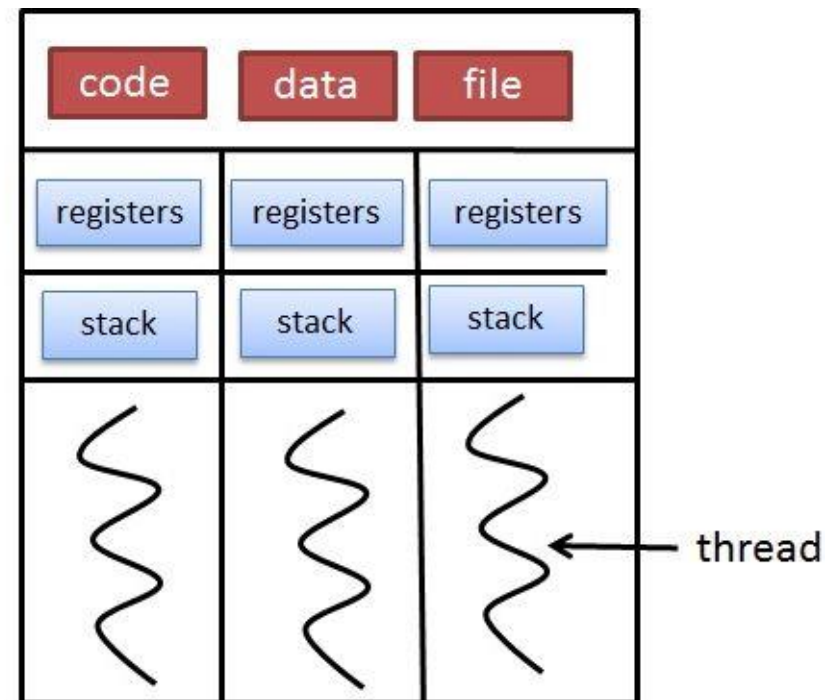


Multithreaded Process

Процессы и потоки

Процесс:

- По процессу на экземпляр программы
- **+1**, если **ещё один экземпляр** программы
- У каждого своё **адресное пространство**
- ...и набор открытых файлов
- ...и соединений, и так далее
- **Изолированы** друг от друга
- ...но есть **shared memory**

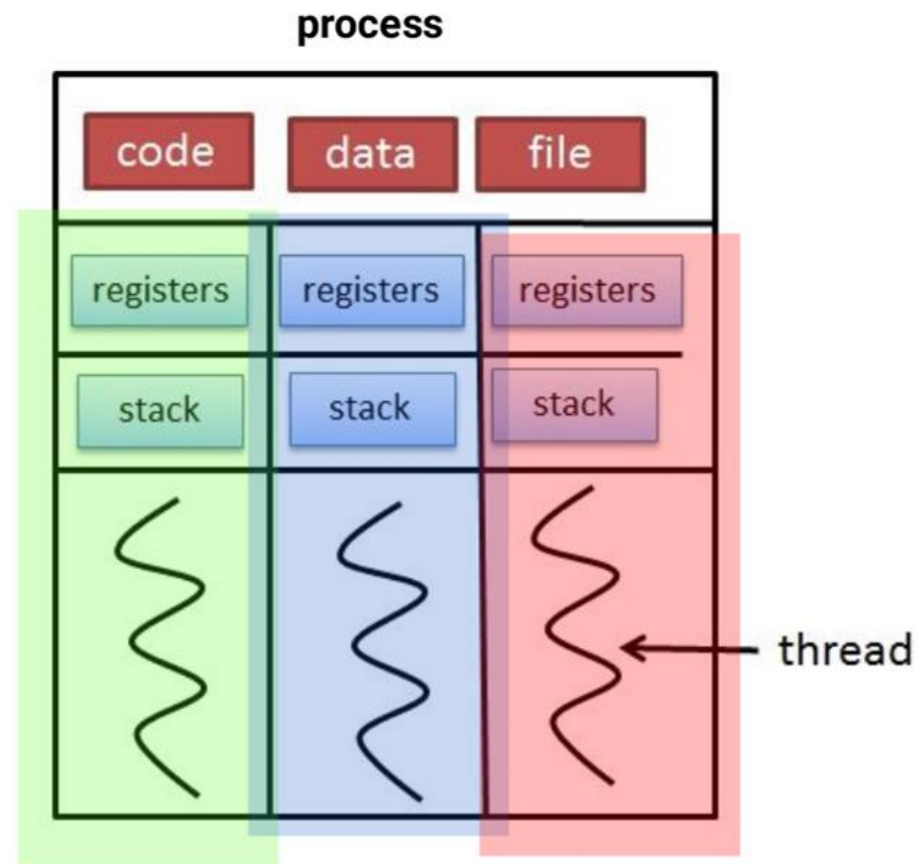


Multithreaded Process

Процессы и потоки

Поток:

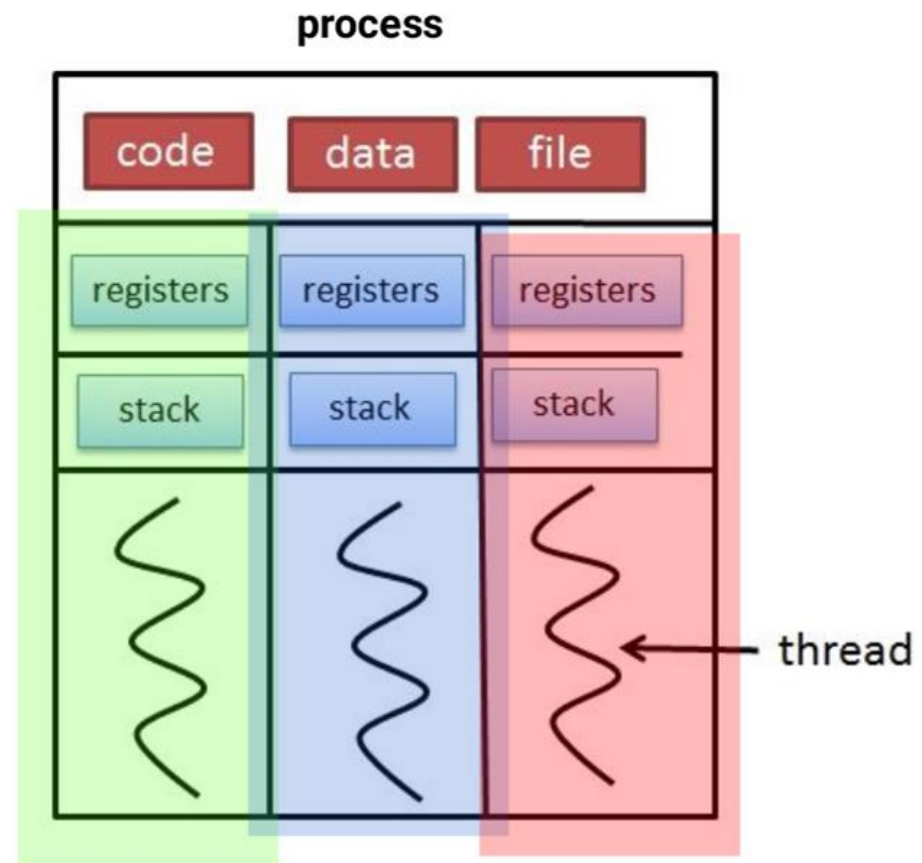
- **Для пользователя** - последовательность инструкций, выполняющихся параллельно, без предписанного порядка во времени



Процессы и потоки

Поток:

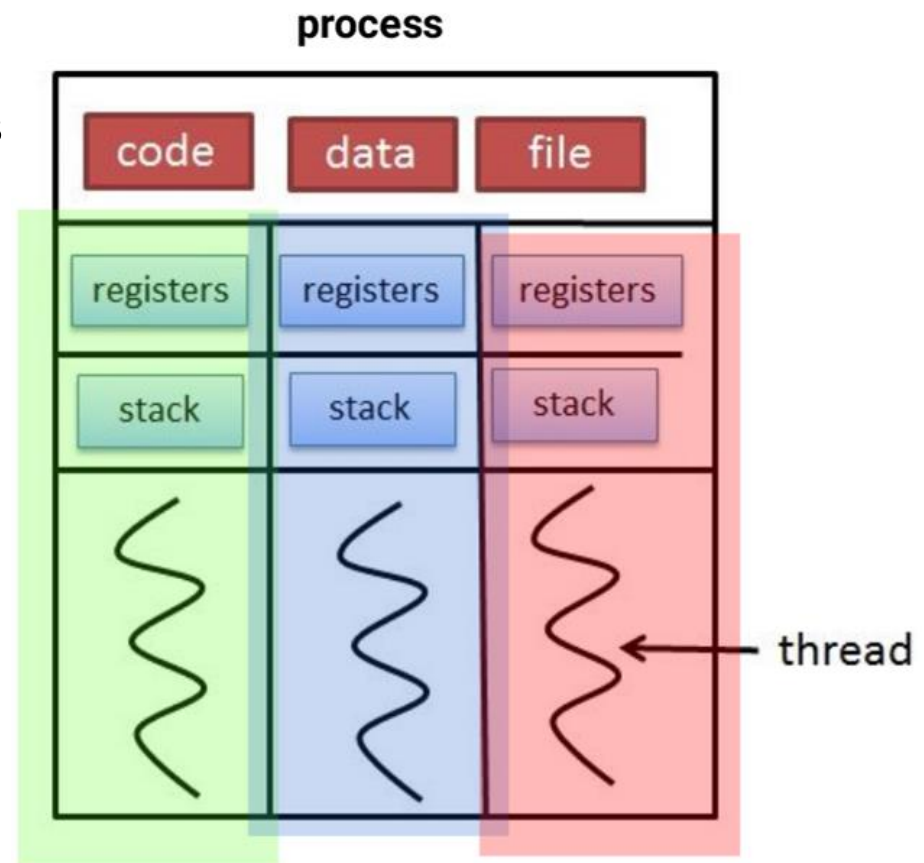
- **Для пользователя** - последовательность инструкций, выполняющихся параллельно, без предписанного порядка во времени
- **Для ОС** - контейнер,
в котором хранятся:
 - счётчик команд
 - регистры процессора
 - стек



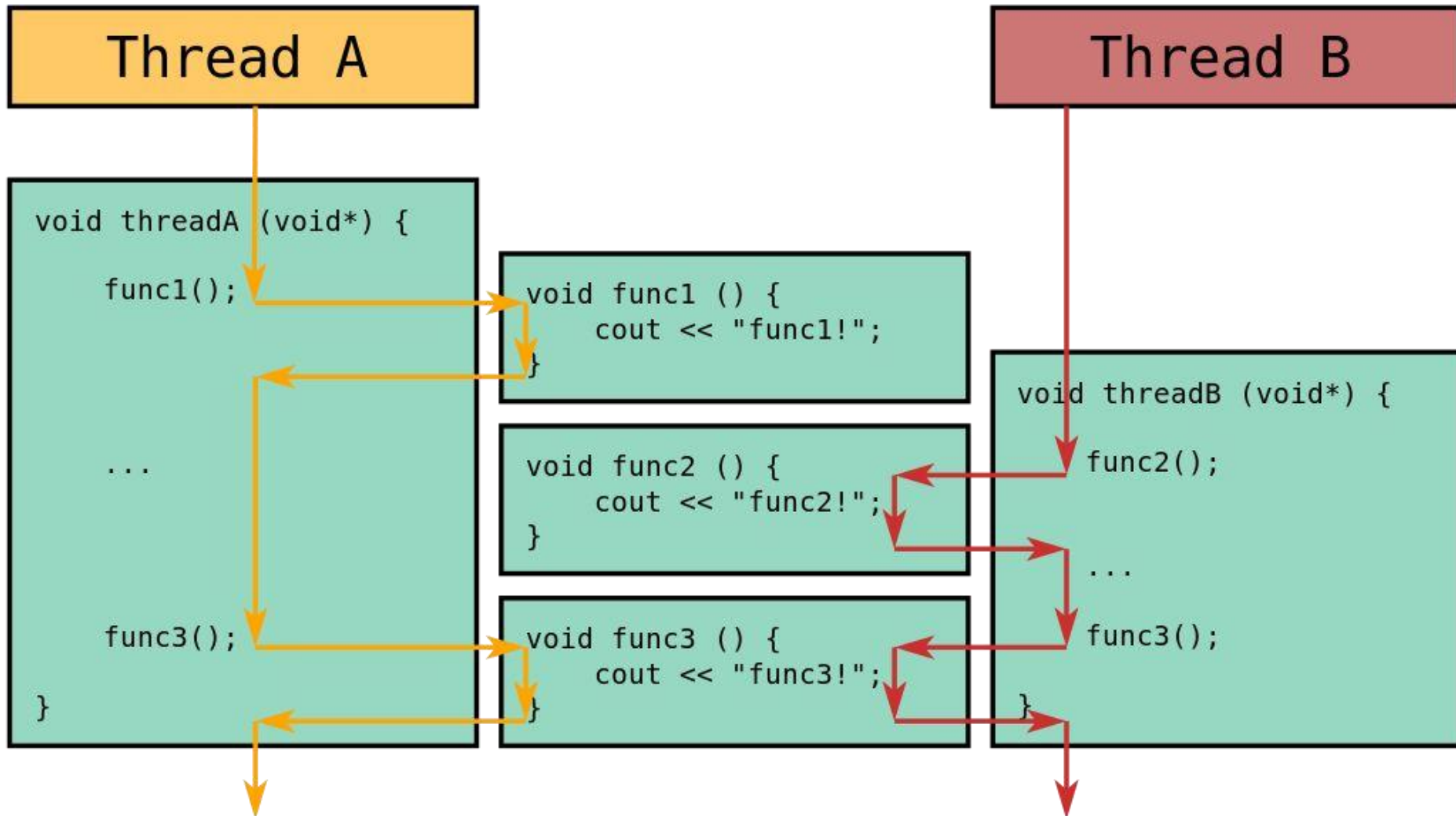
Процессы и потоки

Поток:

- [1.. N] в процессе
- Все в одном **своём процессе**
- У каждого **свой стек вызовов**
- У каждого **свои регистры**
- Каждый имеет доступ к **адресному пространству** процесса
- **Не изолированы** друг от друга внутри процесса



Поток – путешествие во времени



Поток – много их

Все процессы содержат
больше потоков:

- чем процессоров
на платформе
- чем всех ядер всех
процессоров

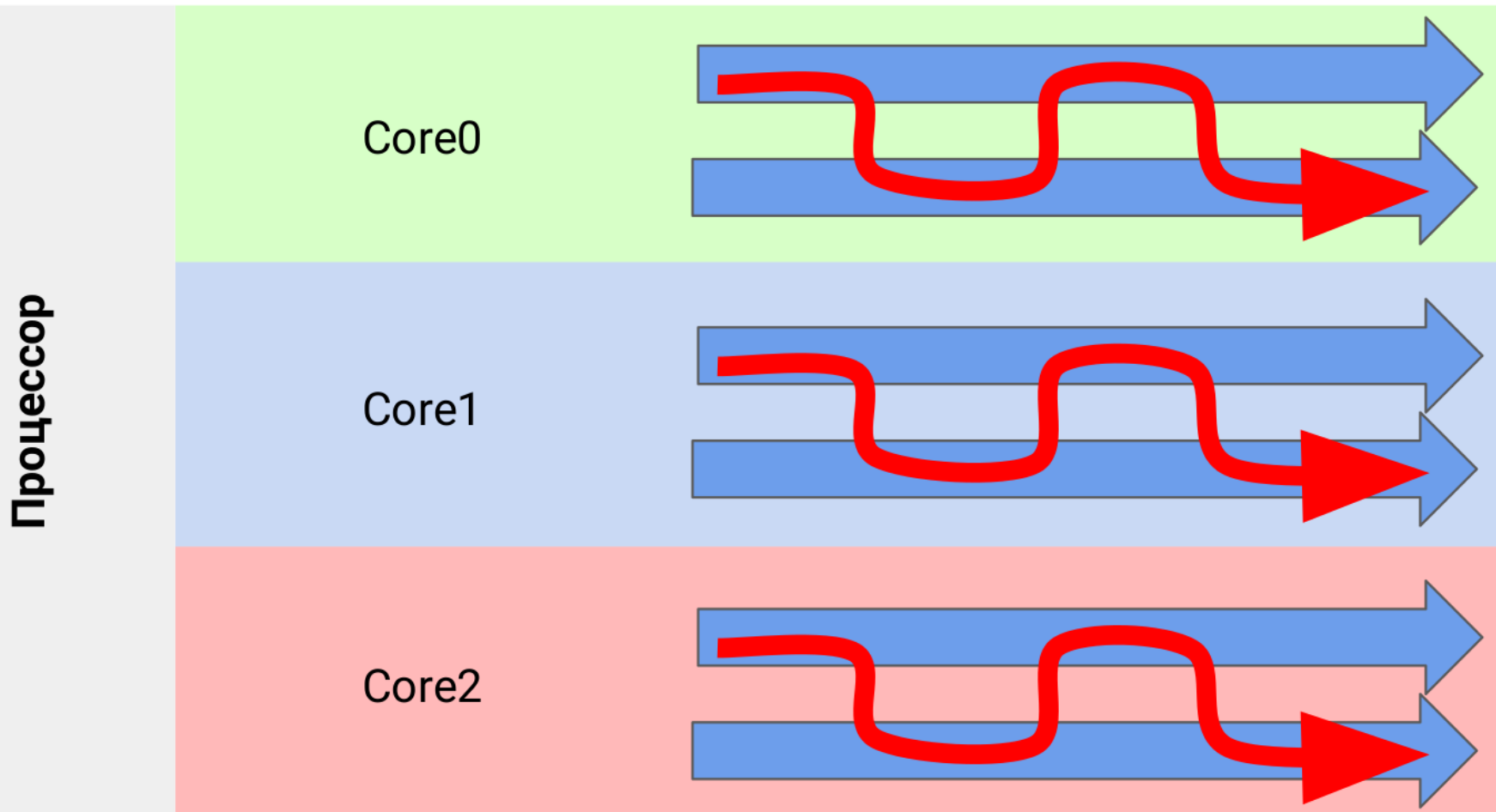
Вас много – я одна!



www.mastertext.spb.ru

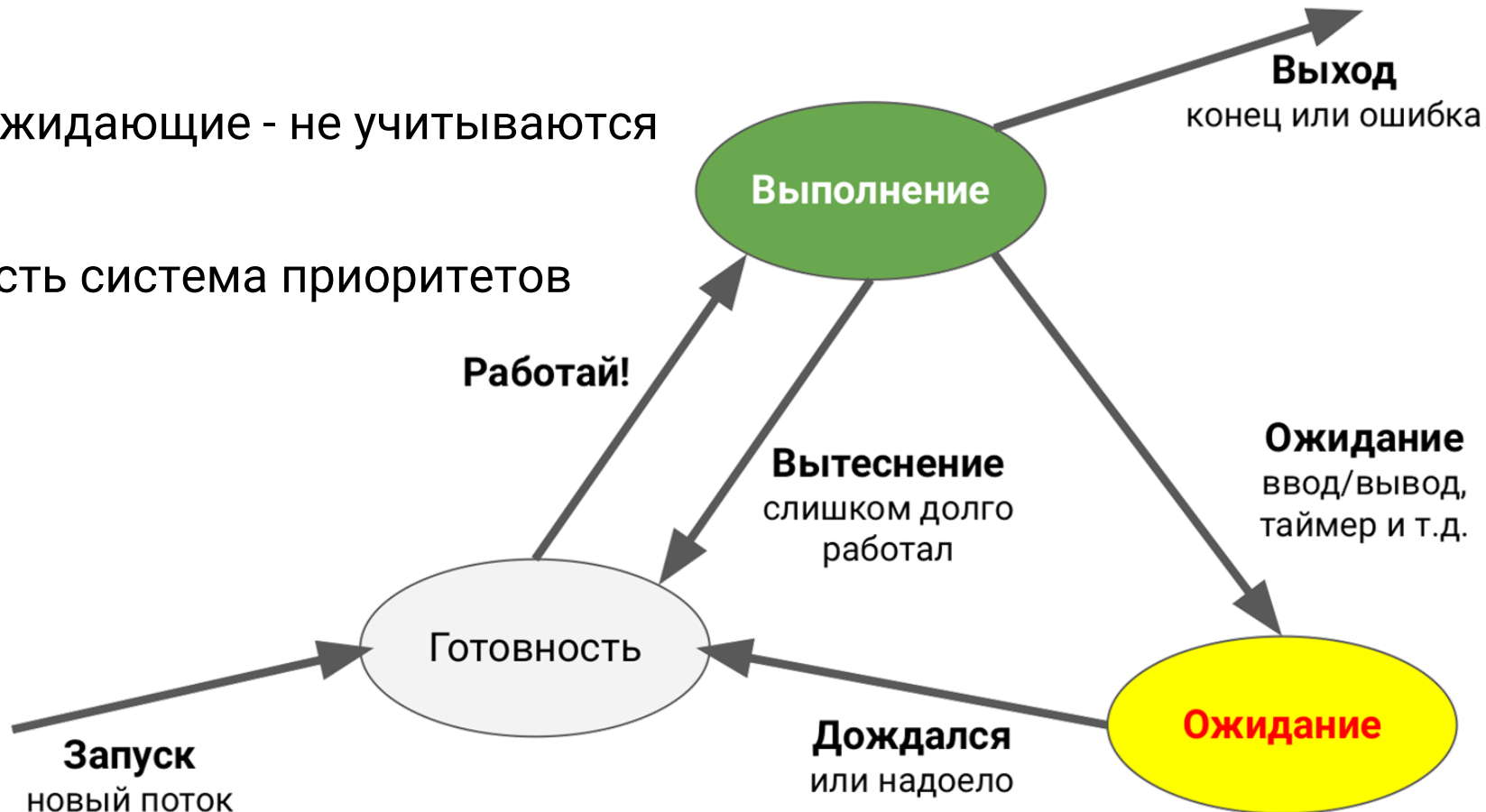
Поток – много их

Имитация одновременной работы путем быстрого переключения



Планирование и диспетчеризация потоков

- Каждому выделяется квант времени
- Ожидающие - не учитываются
- Есть система приоритетов

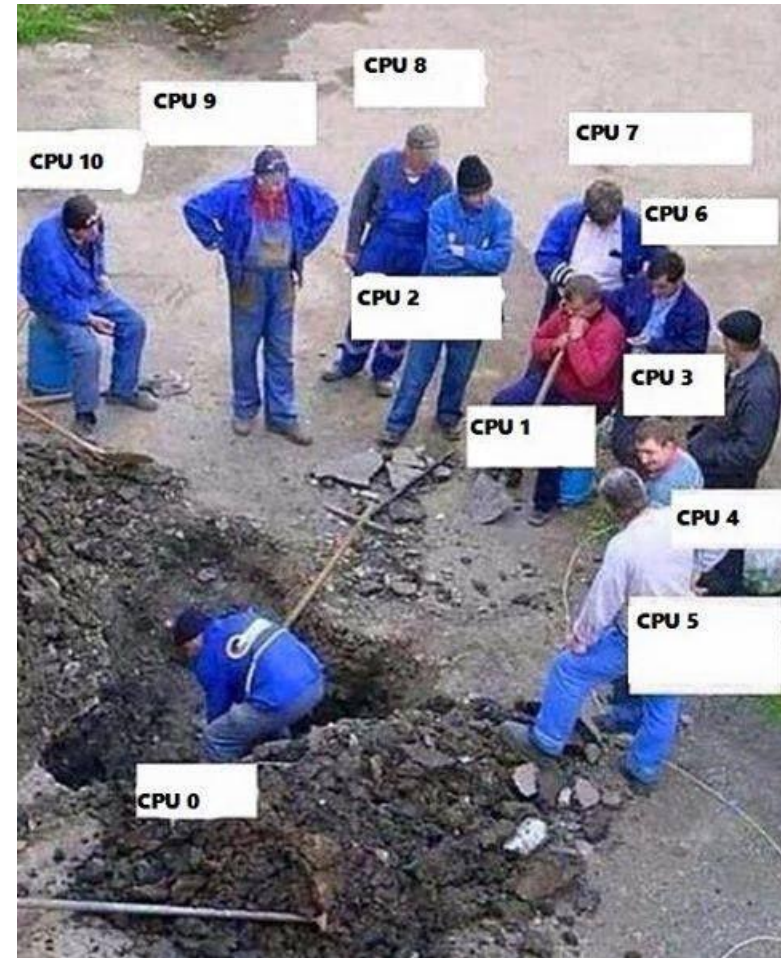


Планирование и диспетчеризация потоков

- **Планирование** – выбор момента времени и потока для выполнения следующим
 - динамическое - по алгоритму, на основе текущей ситуации
 - статическое - predetermined заранее
- **Диспетчеризация** - процедура переключения на выбранный поток. Переключение:
 - сохранение контекста текущего потока
 - загрузка контекста нового потока
 - запуск нового потока на выполнение

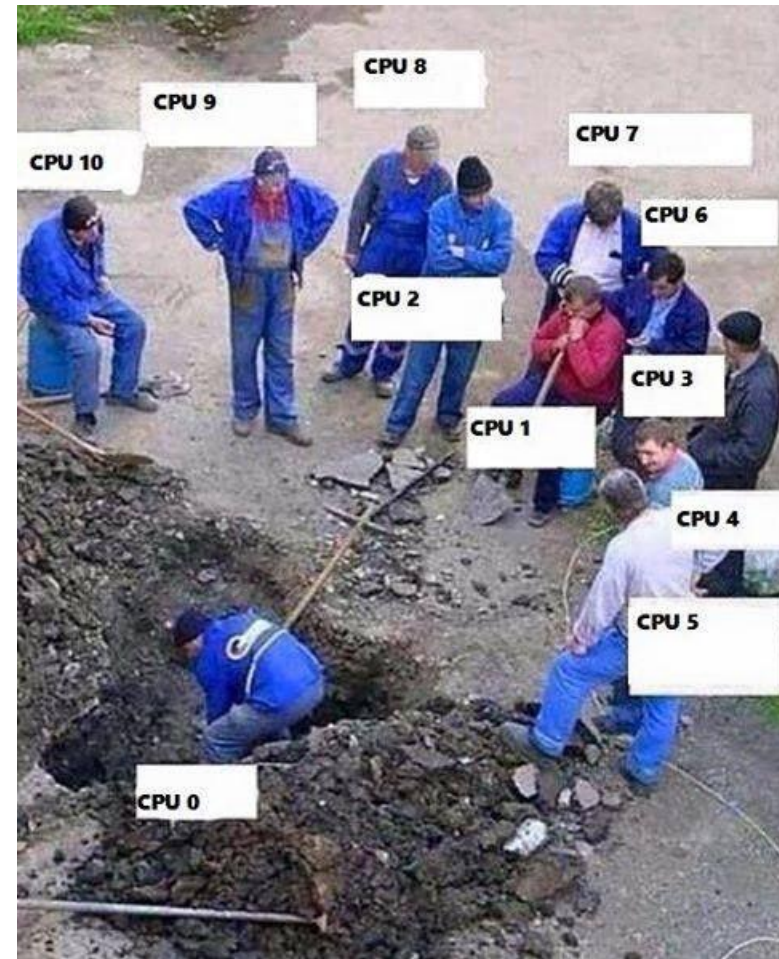
Зачем многопоточно?

- **Декомпозиция** задачи на потоки выполнения упрощает структуру



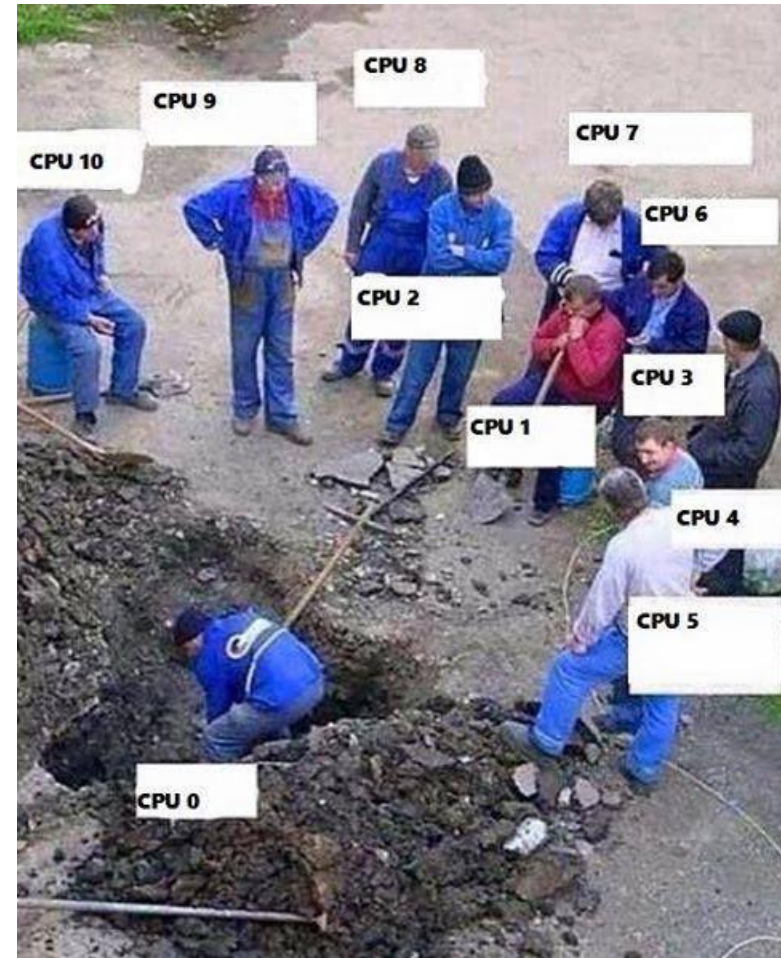
Зачем многопоточно?

- **Декомпозиция** задачи на потоки выполнения упрощает структуру
- Повышение общей **производительности** – не ждать всех медленных задач



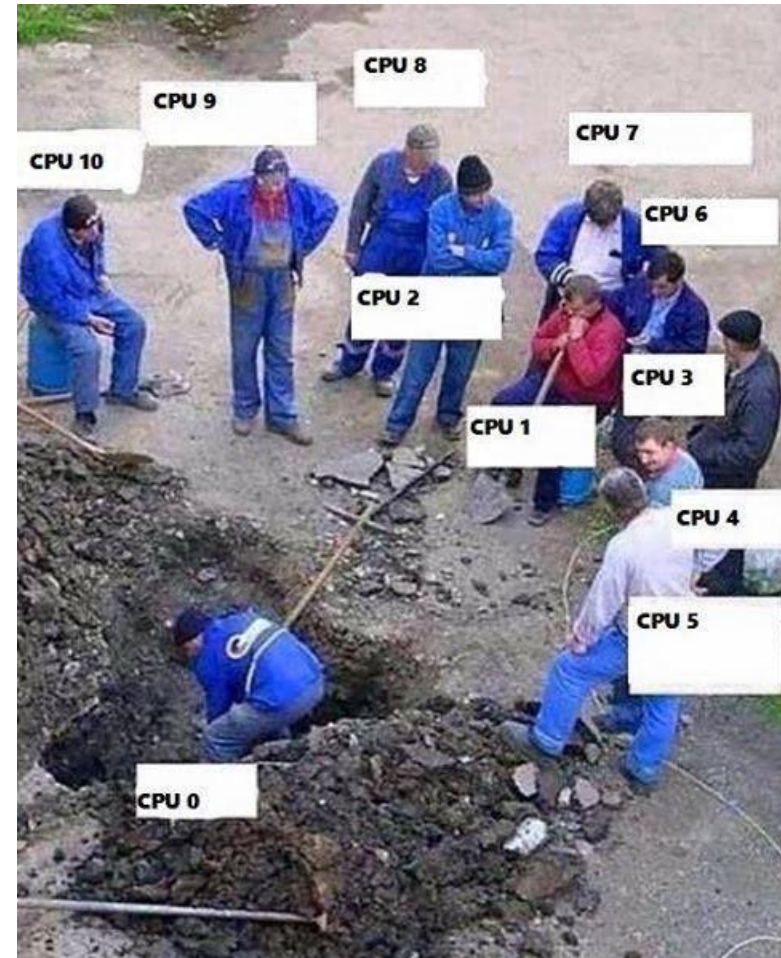
Зачем многопоточно?

- **Декомпозиция** задачи на потоки выполнения упрощает структуру
- Повышение общей **производительности** – не ждать всех медленных задач
- **Операции ввода-вывода** асинхронные I/O от ОС или свои



Зачем многопоточно?

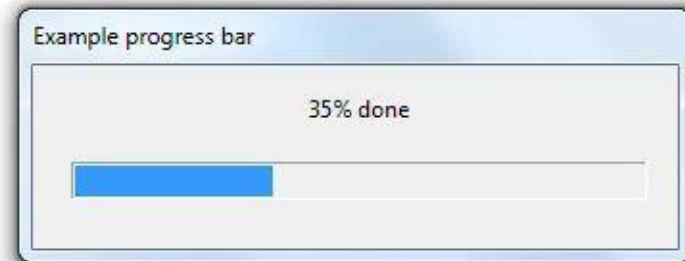
- **Декомпозиция** задачи на потоки выполнения упрощает структуру
- Повышение общей **производительности** – не ждать всех медленных задач
- **Операции ввода-вывода** асинхронные I/O от ОС или свои
- **Масштабирование** – параллельные вычисления



Многопоточно - где можно?

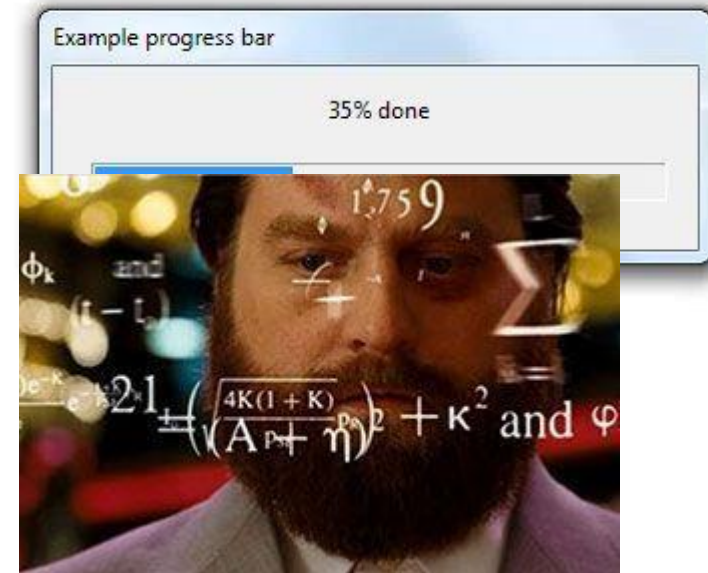
- **Пользовательский интерфейс**

UI – один поток, подготовка данных – другие



Многопоточно - где можно?

- **Пользовательский интерфейс**
UI – один поток, подготовка данных – другие
- **Тяжелые вычисления**
видео, изображения, математика и т.д.



Многопоточно - где можно?

- **Пользовательский интерфейс**
UI – один поток, подготовка данных – другие
- **Тяжелые вычисления**
видео, изображения, математика и т.д.
- **I/O, серверы и клиенты**
web-серверы, браузеры, работа с FS



Многопоточно - где можно?

- **Пользовательский интерфейс**
UI – один поток, подготовка данных – другие
- **Тяжелые вычисления**
видео, изображения, математика и т.д.
- **I/O, серверы и клиенты**
web-серверы, браузеры, работа с FS
- **Везде**



std::thread

Запуск потока

- Класс `std::thread` используется для управления потоками
- При создании объекта с **непустым конструктором** запускается **новый поток**
- Конструктор позволяет передать **callable** объект:
 - указатель на функцию
 - lambda'y
 - функциональный объект
 - ... и произвольный список аргументов с помощью [parameter pack](#)

```
template<typename Function, typename... Args>  
explicit thread(Function &&f, Args &&... args );
```

Ожидание завершения потока

- Поток завершится, когда **завершится запущенная в нем функция**
- Для **ожидания завершения** потока можно вызвать метод **join()**
- Если поток не завершен и не отсоединен, то это трактуется как ошибка и будет вызван **std::terminate**
- Можно использовать RAII-обертку, чтобы избежать описанной выше ситуации. Пример: `thread_guard.cpp`

Отсоединение потока

- Если нужно - то его можно **перевести в фоновый режим** (отсоединить, не ждать его завершения)
`std::thread::detach()`
- Отсоединенный поток не может быть присоединен вновь
- Отсоединить можно только не завершившийся поток
- Пример: долгая операция сохранения данных в файл
`savefile_bg.cpp`

Передача аргументов функции потока

- Все аргументы переданные функции потоку **копируются**
...даже если функция потока **принимает ссылки**
- Чтобы передать ссылку нужно использовать обертки:
 - `std::ref()` для ссылок
 - `std::cref()` для константных ссылок
- При передаче ссылки в поток важно гарантировать, что, время жизни объекта было не меньше, чем время работы потока
- Пример: долгая операция сохранения данных в файл с передачей ссылки на данные `savefile_bg_cref.cpp`

Вспомогательные инструменты

- Определить число аппаратных потоков -
`std::thread::hardware_concurrency()`
- Получить идентификатор для работы с системной библиотекой потоков – `std::thread::native_handle()`:
 - pthread для Linux и MacOS
 - WIN32 API
- Усыпить текущий поток - `std::this_thread::sleep_for()`
- Сигнал для планировщика передать ресурс процессора другому потоку – `std::this_thread::yield()`

Заполните, пожалуйста, опрос о занятии

Заключение

Вопросы?