

Яндекс Такси

Незаменимый C++

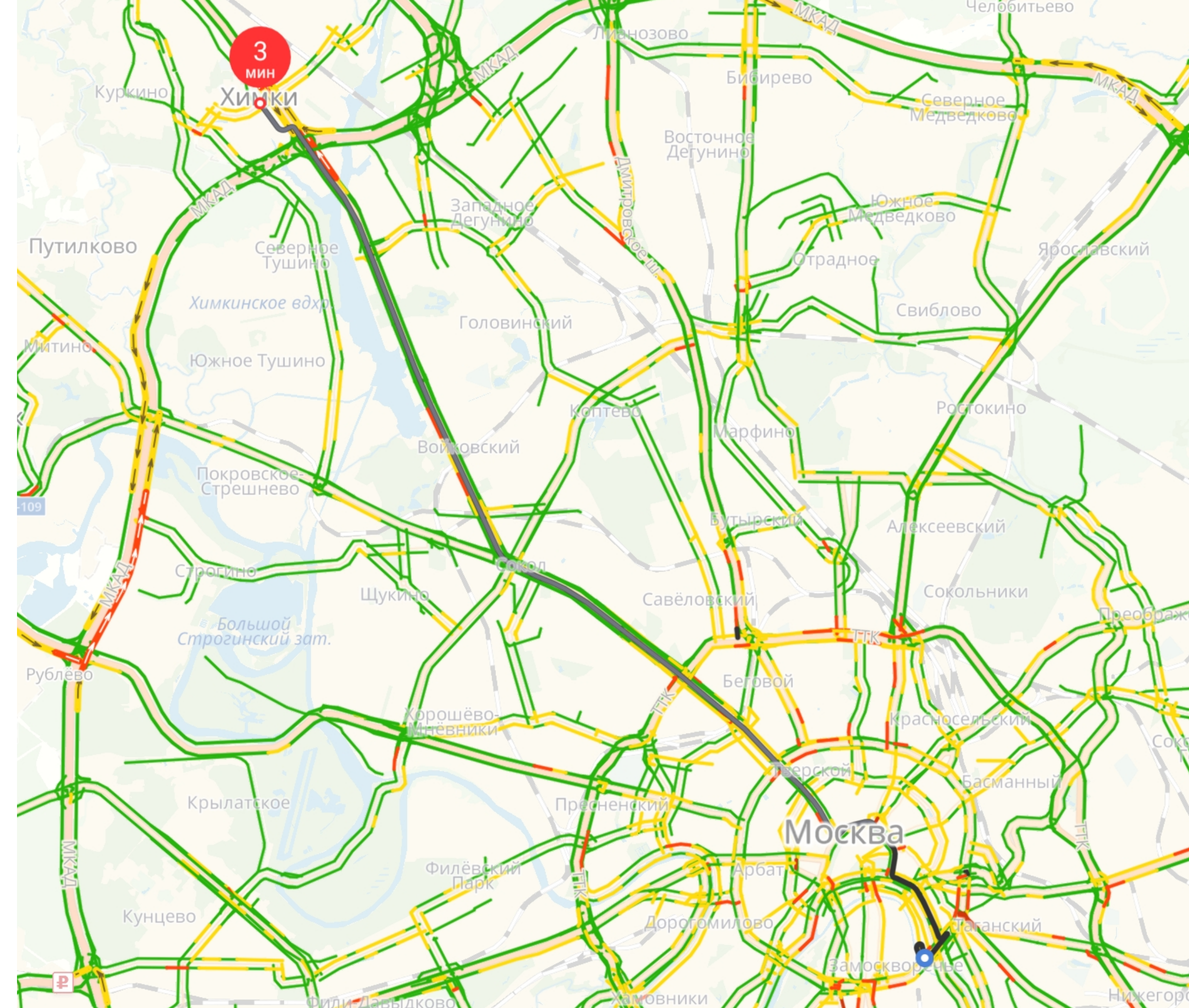
Полухин Антон

Antony Polukhin

Яндекс Такси

Содержание

- На C++ больше не пишут?
- C++ vs. Asm
- C++ vs. Rust
- C++ vs. Go
- C++ vs. (Java + C#)
- Слабые места C++
- Что с этим делать



Asm

Подъезд



C++20



ЭКОНОМ
4₽



КОМФОРТ
8₽



КОМФОРТ+
9₽



БИЗНЕС
34₽



МИНИВЭН
15₽



ДЕТСКИЙ
2₽

Комментарий, пожелания

Способ оплаты
Команда Яндекс.Такси

Заблуждение №1

На C++ больше не пишут программ

Программы

Программы

- Поисковые движки

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация
- Компиляторы (не только компиляторы для C++)

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация
- Компиляторы (не только компиляторы для C++)
- «Виртуальные машины»

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация
- Компиляторы (не только компиляторы для C++)
- «Виртуальные машины»
- Научные программы (CERN и Бозон Хиггса)

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация
- Компиляторы (не только компиляторы для C++)
- «Виртуальные машины»
- Научные программы (CERN и Бозон Хиггса)
- Части ОС (Драйверы, userspace)

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация
- Компиляторы (не только компиляторы для C++)
- «Виртуальные машины»
- Научные программы (CERN и Бозон Хиггса)
- Части ОС (Драйверы, userspace)
- Автопром

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация
- Компиляторы (не только компиляторы для C++)
- «Виртуальные машины»
- Научные программы (CERN и Бозон Хиггса)
- Части ОС (Драйверы, userspace)
- Автопром
- Заводы

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация
- Компиляторы (не только компиляторы для C++)
- «Виртуальные машины»
- Научные программы (CERN и Бозон Хиггса)
- Части ОС (Драйверы, userspace)
- Автопром
- Заводы
- Биржа

Программы

- Поисковые движки
- Большинство высоконагруженных программ (Такси)
- Игры
- Браузеры
- Спецэффекты и анимация
- Компиляторы (не только компиляторы для C++)
- «Виртуальные машины»
- Научные программы (CERN и Бозон Хиггса)
- Части ОС (Драйверы, userspace)
- Автопром
- Заводы
- Биржа
- Офисные приложения

ОК, чем C++ привлекает сегодня?

C++

C++

+ zero-overhead

C++

+ zero-overhead

+ неограниченные возможности

C++

- + zero-overhead
- + неограниченные возможности
- + поддержка огромного количества платформ

C++

- + zero-overhead
- + неограниченные возможности
- + поддержка огромного количества платформ
- + безопасность

C++

- + zero-overhead
- + неограниченные возможности
- + поддержка огромного количества платформ
- + безопасность?

C++

- + zero-overhead
- + неограниченные возможности
- + поддержка огромного количества платформ
- + безопасность?
- + небольшой рантайм

C++

- + zero-overhead
- + неограниченные возможности
- + поддержка огромного количества платформ
- + безопасность?
- + небольшой рантайм
- + стабильность

Сравним с другими языками!

C++ *vs.* Asm

C++ vs. Asm

C++ vs. Asm

+ Ассемблер позволяет выжать максимум из железа

C++ vs. Asm

+ Ассемблер позволяет выжать максимум из железа

- непереносимый код

C++ vs. Asm

- + Ассемблер позволяет выжать максимум из железа
- непереносимый код
- очень медленная разработка

C++ vs. Asm

- + Ассемблер позволяет выжать максимум из железа
- непереносимый код
- очень медленная разработка
- не всегда быстрее

C++ vs. Asm

- + Ассемблер позволяет выжать максимум из железа
- непереносимый код
- очень медленная разработка
- **не всегда быстрее**

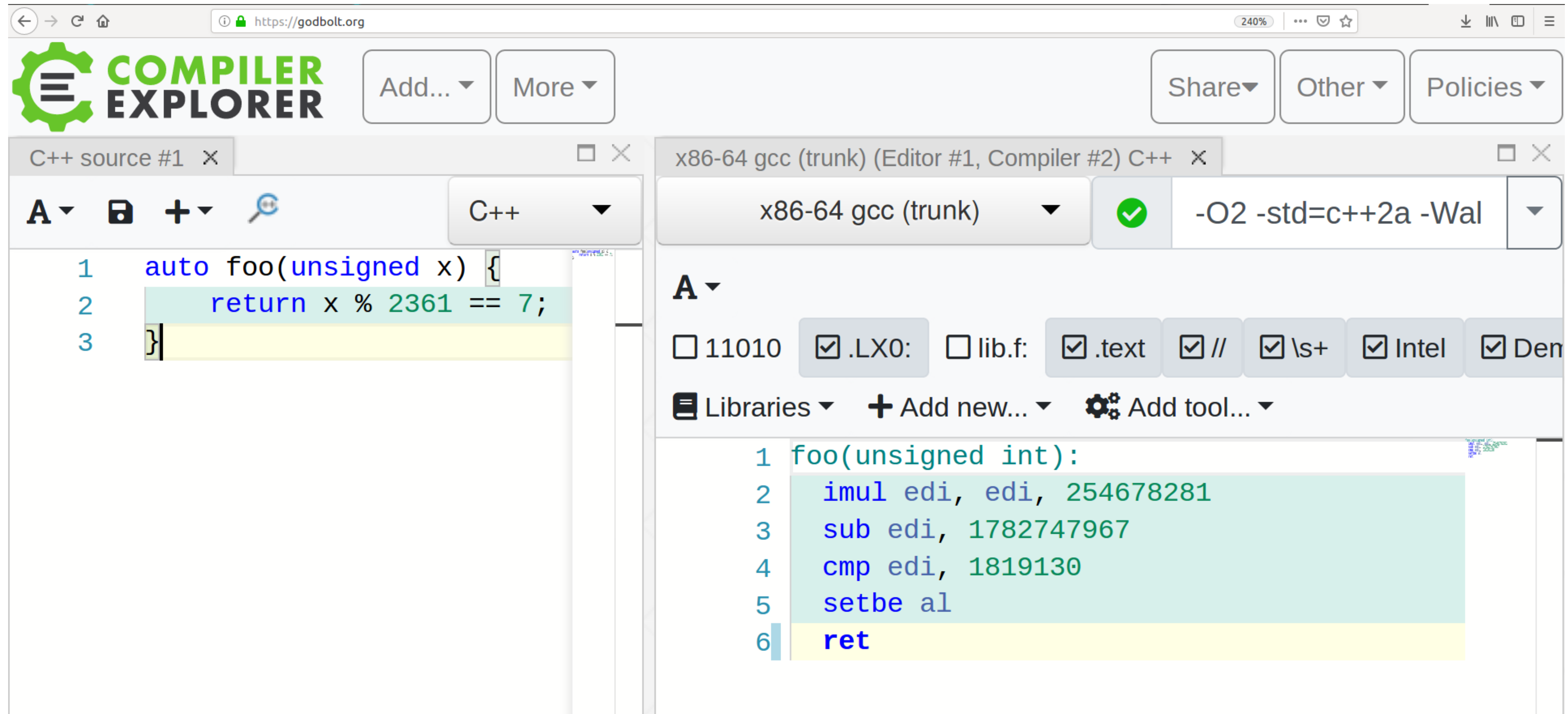
Заблуждение №2

Большие программы на C++ работают медленнее чем на ASM

C++ vs. Asm (или как завалить Asm разработчика)

$x \% 2361 == 7$

C++ vs. Asm (или как завалить Asm разработчика)



C++ vs. Asm (или как завалить Asm разработчика)

```
11750      unsigned).
11751      (X * C3) + C4 <= 2 * C4, where
11752      C3 is modular multiplicative inverse of (unsigned) C1 and 1<<prec and
11753      C4 is ((1<<(prec - 1) - 1) / C1).
11754      If C1 is even, S = ctz(C1), use
11755      ((X * C3) + C4) r>> S <= (C4 >> (S - 1))
11756      where C3 is modular multiplicative inverse of (unsigned)(C1>>S) and 1<<prec
11757      and C4 is ((1<<(prec - 1) - 1) / (C1>>S)) & (-1<<S).
11758
11759      See the Hacker's Delight book, section 10-17.  */
11760      enum tree_code
11761      maybe_optimize_mod_cmp (enum tree_code code, tree *arg0, tree *arg1)
11762      {
```

C++ *vs.* Rust

C++ vs. Rust

C++ vs. Rust

+ Отличная безопасность?

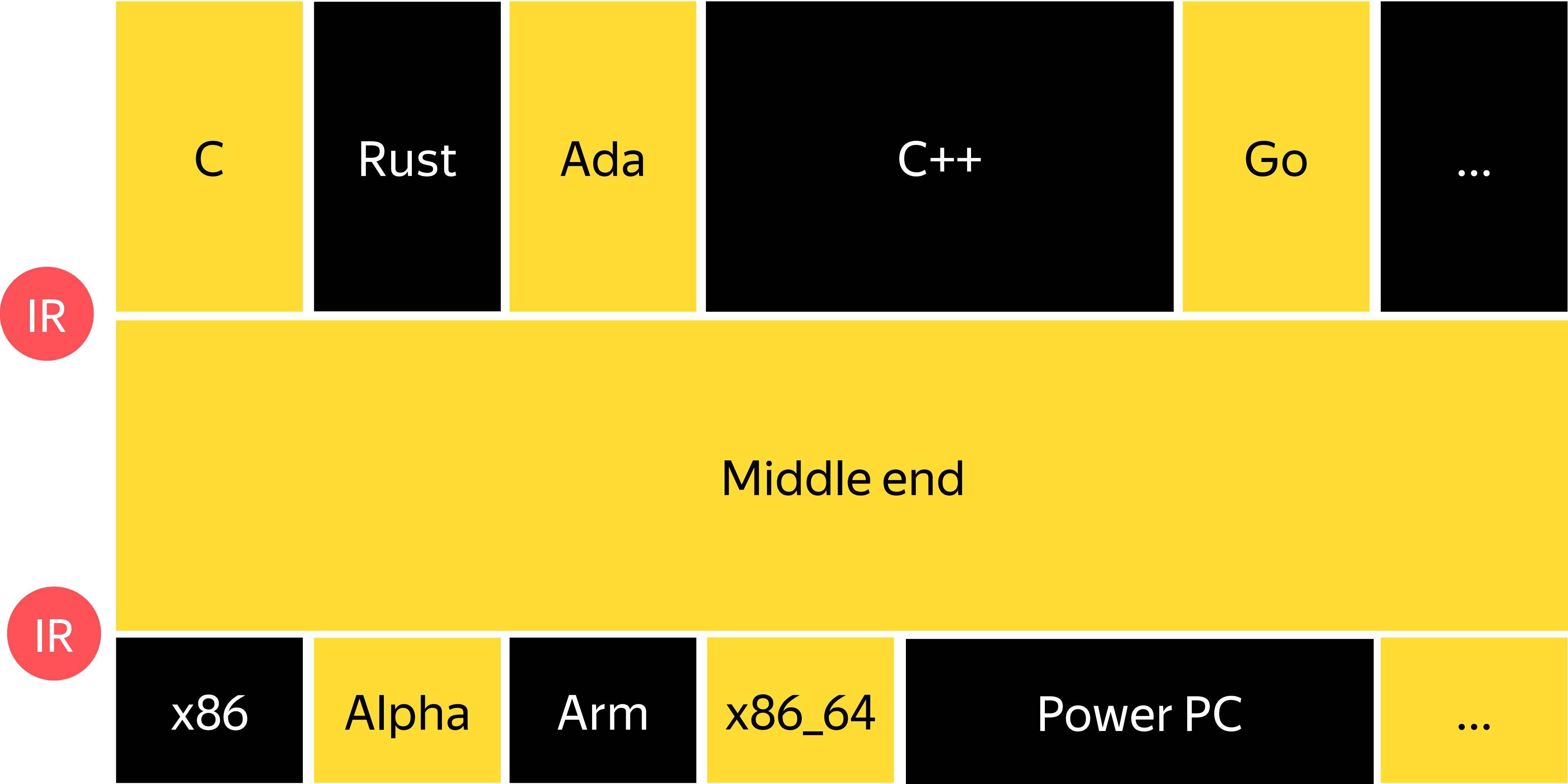
C++ vs. Rust

- + Отличная безопасность?
- + Потенциальная возможность оптимизировать лучше C++

Заблуждение №3

Rust более/такой же производительный как C++

Анатомия компилятора (упрощённо)



C++ vs. Rust

Rust source #2 × rustc nightly (Editor #2, Compiler #2) Rust ×

A ▾ Save/Load + Add new... ▾ Rust ▾

1

2

3 pub fn square(num: i32) -> i32 {

4 return num * num

5 }

C++ source #1 × x86-64 clang (trunk) (Editor #1, Compiler #1) C++ ×

A ▾ Save/Load + Add new... ▾ CppInsights C++ ▾

1 #include <stdint>

2

3 auto square(std::int32_t num) {

4 return num * num;

5 }

Diff rustc nightly vs x86-64 clang (trunk) ×

A ▾ Left: rustc nightly -O ▾ Right: x86-64 clang (trunk) -O2 -s... ▾

1 - example::square:

2 mov eax, edi

3 imul eax, edi

4 ret

1 + square(int): # @square(int)

2 mov eax, edi

3 imul eax, edi

4 ret

Где UB ?

Rust source #2 × rustc nightly (Editor #2, Compiler #2) Rust ×

A ▾ Save/Load + Add new... ▾ Rust ▾

1

2

3 pub fn square(num: i32) -> i32 {

4 return num * num

5 }

C++ source #1 × x86-64 clang (trunk) (Editor #1, Compiler #1) C++ ×

A ▾ Save/Load + Add new... ▾ CppInsights C++ ▾

1 #include <stdint>

2

3 auto square(std::int32_t num) {

4 return num * num;

5 }

Diff rustc nightly vs x86-64 clang (trunk) ×

A ▾ Left: rustc nightly -O ▾ Right: x86-64 clang (trunk) -O2 -s... ▾

1 - example::square:

2 mov eax, edi

3 imul eax, edi

4 ret

1 + square(int): # @square(int)

2 mov eax, edi

3 imul eax, edi

4 ret

C++ vs. Rust

Rust source #2 × rustc nightly (Editor #2, Compiler #2) Rust ×

A ▾ Save/Load + Add new... ▾ Rust ▾

1

2

3 pub fn foo(max: i32, num: i32) -> i32 {

4 return max * num

5 }

6

7 pub fn bar(max: i32, num: i32) -> i32 {

8 return bar(max, num) * bar(max, num)

9 }

C++ source #1 × x86-64 clang (trunk) (Editor #1, Compiler #1) C++ ×

A ▾ Save/Load + Add new... ▾ CppInsights C++ ▾

1 #include <cstdint>

2

3 auto foo(std::int32_t max, std::int32_t num) {

4 return max * num;

5 }

6

7 std::int32_t bar(std::int32_t max, std::int32_t num

8 return bar(max, num) * bar(max, num);

9 }

Diff rustc nightly vs x86-64 clang (trunk) ×

A ▾ Left: rustc nightly -O ▾ Right: x86-64 clang (trunk) -O2 -s... ▾

1 - example::foo:

2 mov eax, edi

3 imul eax, esi

4 ret

5 -

6 - example::bar:

7 ret

1 + foo(int, int): # @foo(int, int)

2 mov eax, edi

3 imul eax, esi

4 ret

5 + bar(int, int): # @bar(int, int)

6

6 ret

C++ vs. Rust

Rust source #2 × rustc nightly (Editor #2, Compiler #1) Rust ×

A ▾ Save/Load + Add new... ▾ Rust ▾

1

2

3 pub fn foo(max: i32, num: i32) -> i32 {

4 return max * num

5 }

6

7 pub fn bar(max: i32, num: i32) -> i32 {

8 return foo(max, num) * foo(max, num)

9 }

C++ source #1 × x86-64 clang (trunk) (Editor #1, Compiler #2) C++ ×

A ▾ Save/Load + Add new... ▾ CppInsights C++ ▾

1 #include <stdint>

2

3 auto foo(std::int32_t max, std::int32_t num) {

4 return max * num;

5 }

6

7 std::int32_t bar(std::int32_t max, std::int32_t num) {

8 return foo(max, num) * foo(max, num);

9 }

Diff rustc nightly vs x86-64 clang (trunk) ×

A ▾ Left: rustc nightly -O -C overflow... ▾ Right: x86-64 clang (trunk) -O2 -s... ▾

11 -

12 -

13 - example::bar:

14 - push rax

15 mov eax, edi

16 imul eax, esi

17 jo .LBB1_3

18 imul eax, eax

19 - jo .LBB1_2

20 - pop rcx

7 -

8 + bar(int, int): # @bar(int, int)

9 mov eax, edi

10 imul eax, esi

11 jo .LBB1_3

12 imul eax, eax

13 + jo .LBB1_3

C++ vs. Rust

Rust source #2 × rustc nightly (Editor #2, Compiler #1) Rust ×

A ▾ Save/Load + Add new... ▾ Rust ▾

1

2

3 pub fn mutate(num: i32) -> [i32; 6] {

4 let mut numbers = [1i32, num, 3, 4, 5, 6];

5 for (j, elem) in numbers.iter_mut().enumerate()

6 *elem += j as i32

7 }

8 return numbers;

9 }

C++ source #1 × x86-64 clang (trunk) (Editor #1, Compiler #2) C++ ×

A ▾ Save/Load + Add new... ▾ CppInsights C++ ▾

1 include <array>

2

3 uto mutate(std::int32_t num) {

4 std::array<std::int32_t, 6> numbers = {1, num,

5 for (std::size_t j = 0; auto& elem : numbers) {

6 elem += j;

7 ++j;

8 }

9 return numbers;

10

Diff rustc nightly vs x86-64 clang (trunk) ×

A ▾ Left: rustc nightly -O ▾ Right: x86-64 clang (trunk) -O2 -s... ▾

13 - mov dword ptr [rdi], ecx

14 - mov ecx, dword ptr [rsp + 4]

15 - mov dword ptr [rdi + 4], ecx

16 - mov rcx, qword ptr [rsp + 8]

17 - mov qword ptr [rdi + 8], rcx

18 - mov rcx, qword ptr [rsp + 16]

19 - mov qword ptr [rdi + 16], rcx

20 - add rsp, 24

C++ vs. Rust

Rust source #2 × rustc nightly (Editor #2, Compiler #1) Rust ×

A ▾ Save/Load + Add new... ▾ Rust ▾

```
1
2
3 pub struct Stats { x: u32, y: u32, z: u32, }
4
5 pub fn sum(a: &Stats, b: &Stats) -> Stats {
6     return Stats {x: a.x + b.x, y: a.y + b.y, z: a
7 }
```

C++ source #1 × x86-64 clang (trunk) (Editor #1, Compiler #2) C++ ×

A ▾ Save/Load + Add new... ▾ CppInsights C++ ▾

```
1 #include <cstdint>
2
3 struct Stats { std::uint32_t x, y, z; };
4
5 auto sum(const Stats& a, const Stats& b) {
6     return Stats {a.x + b.x, a.y + b.y, a.z + b.z
7 }
```

Diff rustc nightly vs x86-64 clang (trunk) ×

A ▾ Left: rustc nightly -O ▾ Right: x86-64 clang (trunk) -O2 -s... ▾

| | |
|--|--|
| <pre>1 - _ZN7example3sum17h2331b594b0172e4aE: 2 - mov ecx, dword ptr [rdx] 3 - mov r8d, dword ptr [rdx + 4] 4 - add ecx, dword ptr [rsi] 5 - add r8d, dword ptr [rsi + 4] 6 - mov edx, dword ptr [rdx + 8] 7 - add edx, dword ptr [rsi + 8] 8 - mov rax, rdi 9 - mov dword ptr [rdi], ecx 10 - mov dword ptr [rdi + 4], r8d 11 - mov dword ptr [rdi + 8], edx 12 ret</pre> | <pre>1 + sum(Stats const&, Stats const&): # @sum(Stats const&, S 2 + mov eax, dword ptr [rsi] 3 + add eax, dword ptr [rdi] 4 + mov ecx, dword ptr [rsi + 4] 5 + add ecx, dword ptr [rdi + 4] 6 + mov edx, dword ptr [rsi + 8] 7 + add edx, dword ptr [rdi + 8] 8 + shl rcx, 32 9 + or rax, rcx 10 ret</pre> |
|--|--|

C++ vs. Rust

Вроде норм.

C++ vs. Rust

Вроде норм.

Переходим на Rust?

C++ vs. Rust

Вроде норм.

Переходим на Rust?

Oh, wait!..

C++ vs. Rust

Вроде норм.

Переходим на Rust?

Oh, wait!..

$C \rightarrow C++$ — noop

$C \rightarrow \text{Rust}$ — PAIN!!!!!!!

C → Rust

- `unsafe {}` → нет безопасности

C → Rust

- `unsafe {}` → нет безопасности
- UB: <https://doc.rust-lang.org/reference/behavior-considered-undefined.html>

C → Rust

- `unsafe {}` → нет безопасности
- UB: <https://doc.rust-lang.org/reference/behavior-considered-undefined.html>

It is the programmer's responsibility when writing `unsafe` code to ensure that any safe code interacting with the `unsafe` code cannot trigger these behaviors. `unsafe` code that satisfies this property for any safe client is called *sound*; if `unsafe` code can be misused by safe code to exhibit undefined behavior, it is *unsound*.

⚠ **Warning:** The following list is not exhaustive. There is no formal model of Rust's semantics for what is and is not allowed in unsafe code, so there may be more behavior considered unsafe. The following list is just what we know for sure is undefined behavior. Please read the

C → Rust

- `unsafe {}` → нет безопасности
- UB: <https://doc.rust-lang.org/reference/behavior-considered-undefined.html>
- Нет возможности использовать C headers
 - Надо генерировать свои
 - Обновление библиотек — БОЛЬ!
 - Надо headers руками допатчивать
 - Мучительные страдания с borrow checker на сложных C проектах
[<https://hackernoon.com/why-im-dropping-rust-fd1c32986c88>]

Заблуждение №4

Программа написанная на языке ~~Rust~~ X не содержит ошибок

Anything

- unsafe или аналоги → нет безопасности

Anything

- unsafe или аналоги → нет безопасности
- если ваша программа компилируется, это ещё не значит что всё ОК

C++ *vs.* Go

C++ vs. Go

C++ vs. Go

- *<ВЕЛИКОЕ МНОЖЕСТВО>*

C++ vs. Go

- *<великое множество>*

+ Асинхронность и многопоточность на основе
корутин

C++ vs. Go

- *<ВЕЛИКОЕ МНОЖЕСТВО>*

+ Асинхронность и многопоточность на основе корутин

- Boost.Fibers

C++ vs. Go

- *<ВЕЛИКОЕ МНОЖЕСТВО>*

+ Асинхронность и многопоточность на основе корутин

- Boost.Fibers
- Yandex.Taxi userver

C++ vs. Go

- *<ВЕЛИКОЕ МНОЖЕСТВО>*

+ Асинхронность и многопоточность на основе корутин

- Boost.Fibers
- Yandex.Taxi userver
- Quantum

C++ vs. Go

- *<ВЕЛИКОЕ МНОЖЕСТВО>*

+ Асинхронность и многопоточность на основе корутин

- Boost.Fibers
- Yandex.Taxi userver
- Quantum
- Folly fibers

C++ vs. Go

- *<ВЕЛИКОЕ МНОЖЕСТВО>*

+ Асинхронность и многопоточность на основе корутин

- Boost.Fibers
- Yandex.Taxi userver
- Quantum
- Folly fibers
- Coroutines TS

C++ vs. Go

- *<ВЕЛИКОЕ МНОЖЕСТВО>*

+ Асинхронность и многопоточность на основе корутин

- Boost.Fibers
- Yandex.Taxi userver
- Quantum
- Folly fibers
- Coroutines TS
- C++20

Python vs. Go

Go скорее конкурент Python, чем C++

Заблуждение №5

Бенчмарки показывают что программы на X быстрее C++

Добро пожаловать в мир
«честных» бенчмарков!

Типичные ошибки

- Отключается сборщик мусора

Типичные ошибки

- Отключается сборщик мусора
- Код написан не на C++

Типичные ошибки

- Отключается сборщик мусора
- Код на C++ написан в стиле `float* f = new float;`

Типичные ошибки

- Отключается сборщик мусора
- Код на C++ написан в стиле `float* f = new float;`
- На X написана другая программа

Типичные ошибки

- Отключается сборщик мусора
- Код на C++ написан в стиле `float* f = new float;`
- На X написана другая программа
- Тестируется библиотека а не язык

Типичные ошибки

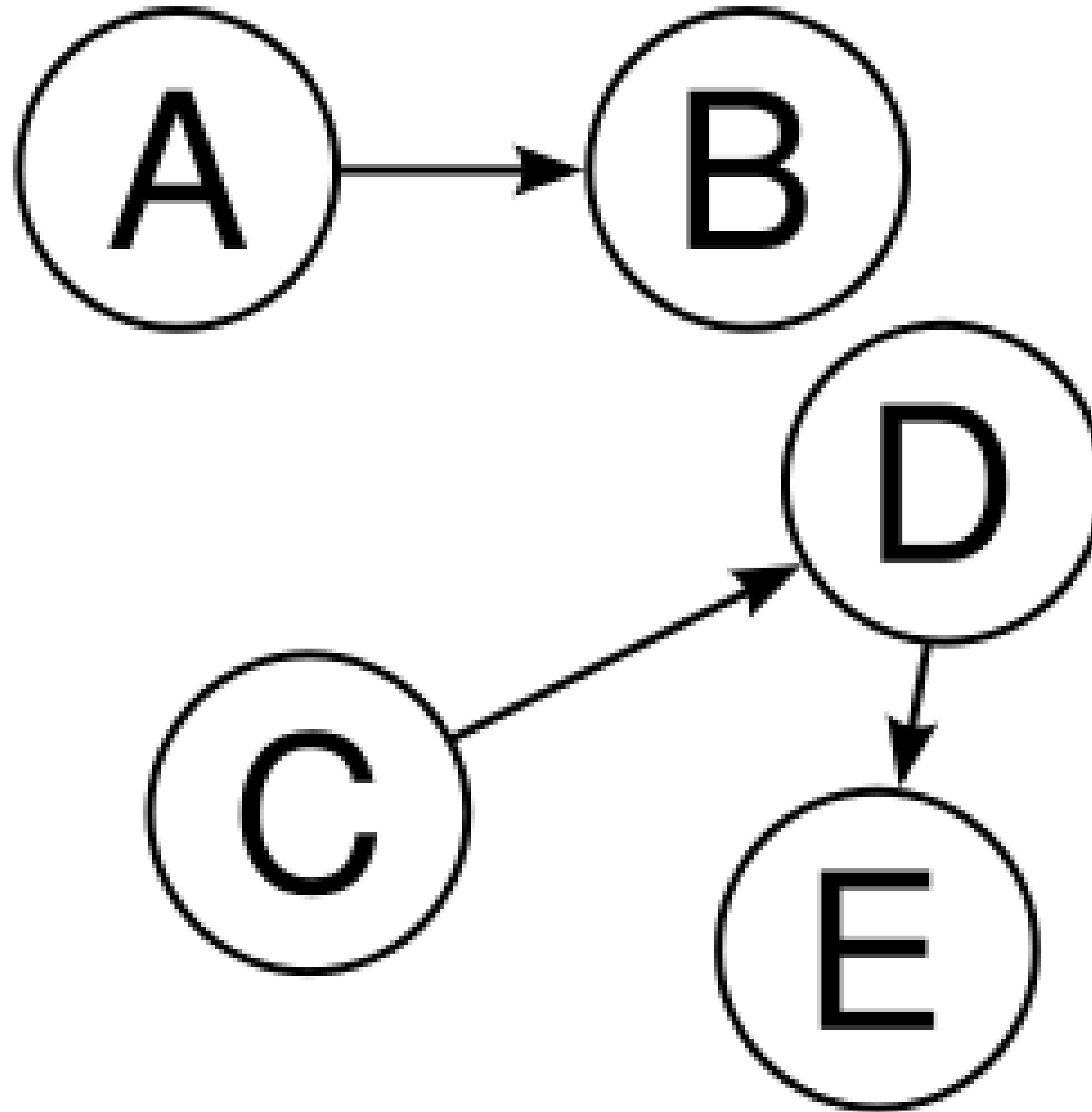
- Отключается сборщик мусора
- Код на C++ написан в стиле `float* f = new float;`
- На X написана другая программа
- Тестируется библиотека а не язык
- Сравниваются разные оптимизаторы, а не языки:

```
_mm_sub_pd(  
    _mm_mul_pd(distance, _mm_set1_pd(1.5)),  
    _mm_mul_pd(  
        _mm_mul_pd(_mm_mul_pd(_mm_set1_pd(0.5), dsquared), distance),  
        _mm_mul_pd(distance, distance),  
    ),  
)
```

Заблуждение №6

Сборщик мусора не добавляет накладных расходов

Mark and sweep



structures

```
struct list_node {  
    list_node* next;  
    list_node* prev;  
};  
  
struct slist_node {  
    slist_node* next;  
};
```

structures

```
vector<void*> root;
```

structures

```
vector<void*> root; // root[0] – это slist_node или list_node?
```

structures

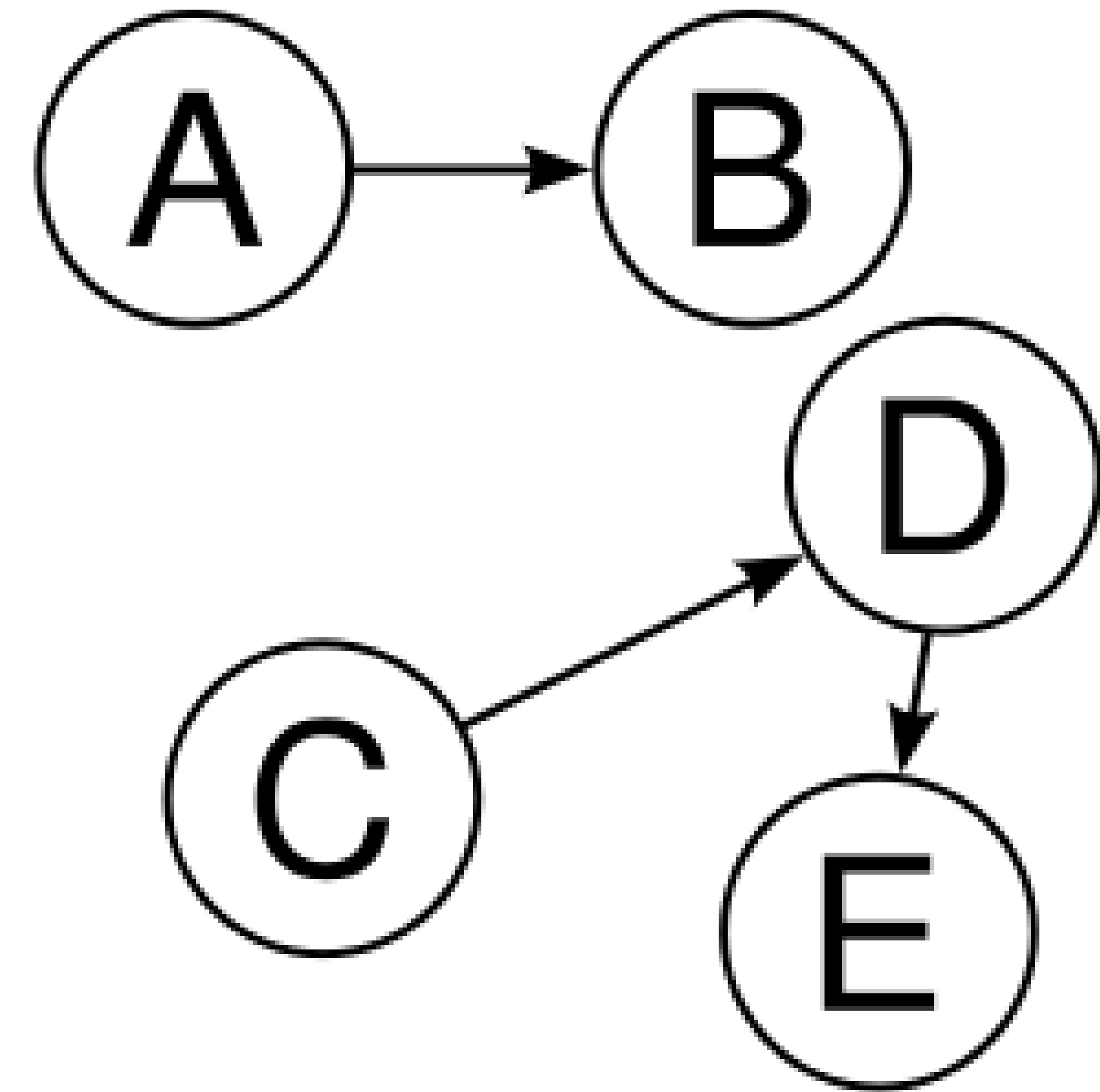
```
vector<void*> root; // Сколько указателей и где они?  
vector<Object*> root;
```

structures

```
struct list_node {  
    __meta vptr;  
    list_node* next;  
    list_node* prev;  
};  
  
struct slist_node {  
    __meta vptr;  
    slist_node* next;  
};
```

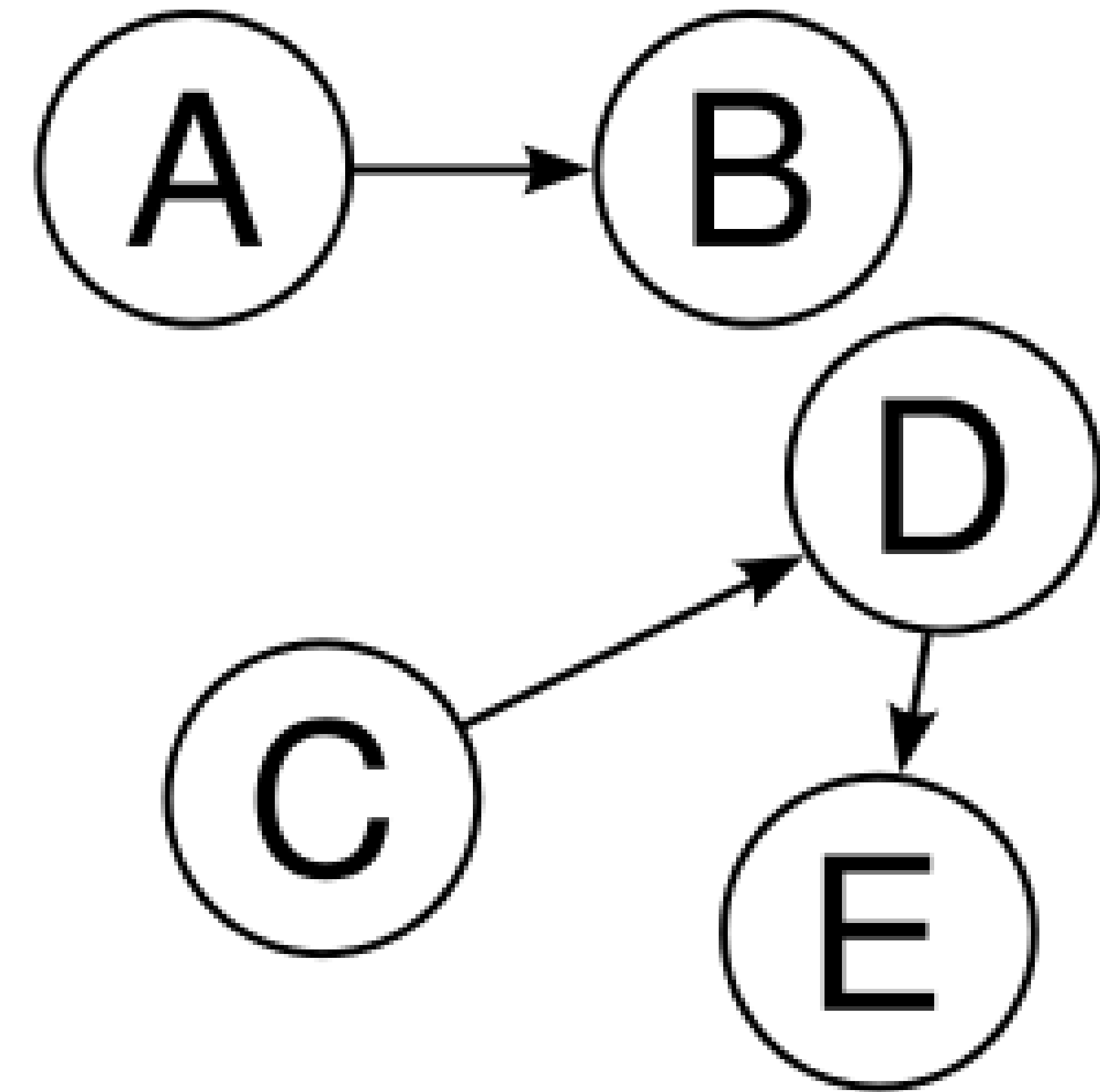

Stop the world

- Все потоки останавливаются



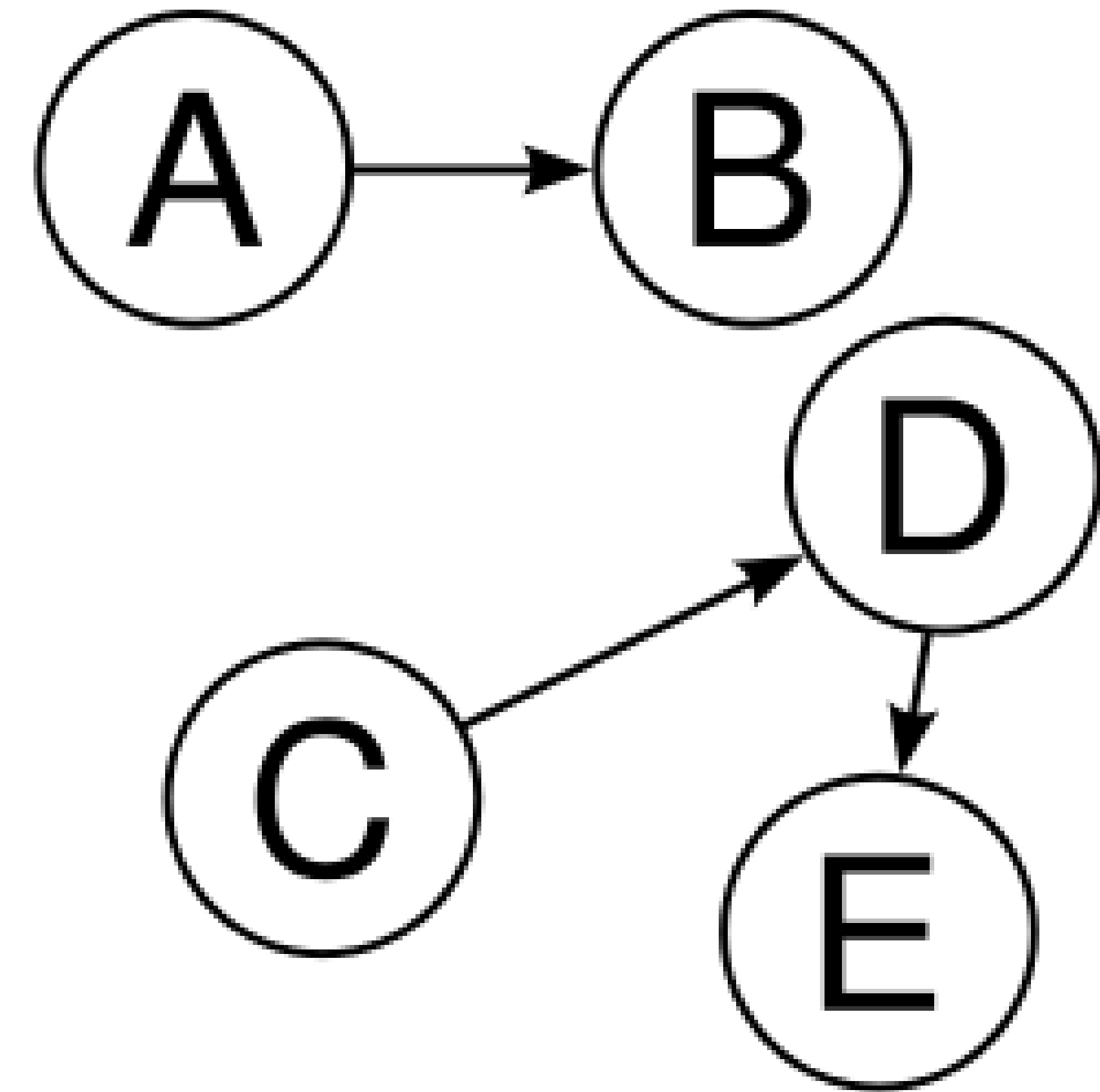
Stop the world

- Все потоки останавливаются
 - Проходим по всем узлам графа, помечая достижимые узлы



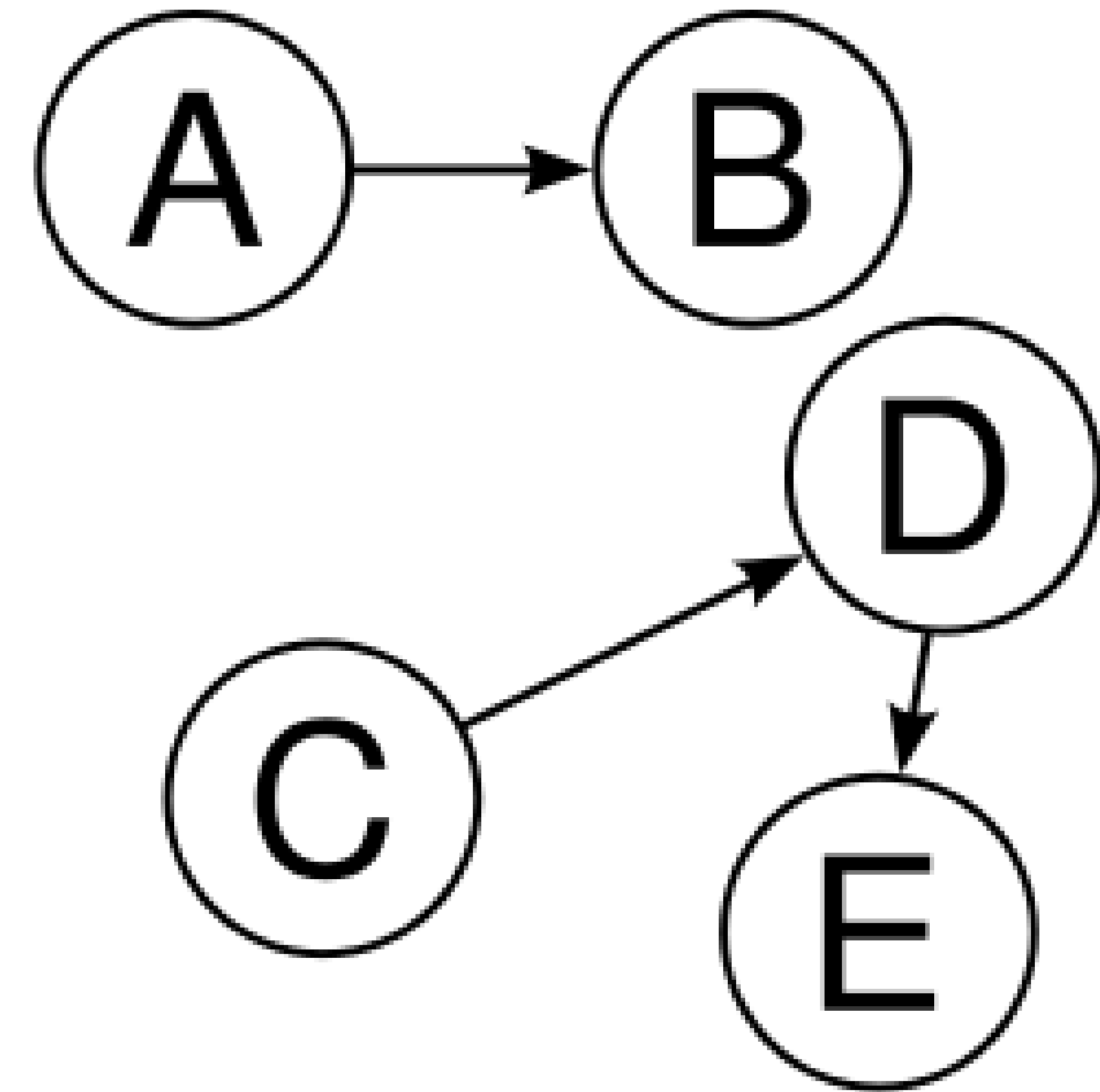
Stop the world

- Все потоки останавливаются
 - Проходим по всем узлам графа, помечая достижимые узлы
 - ????



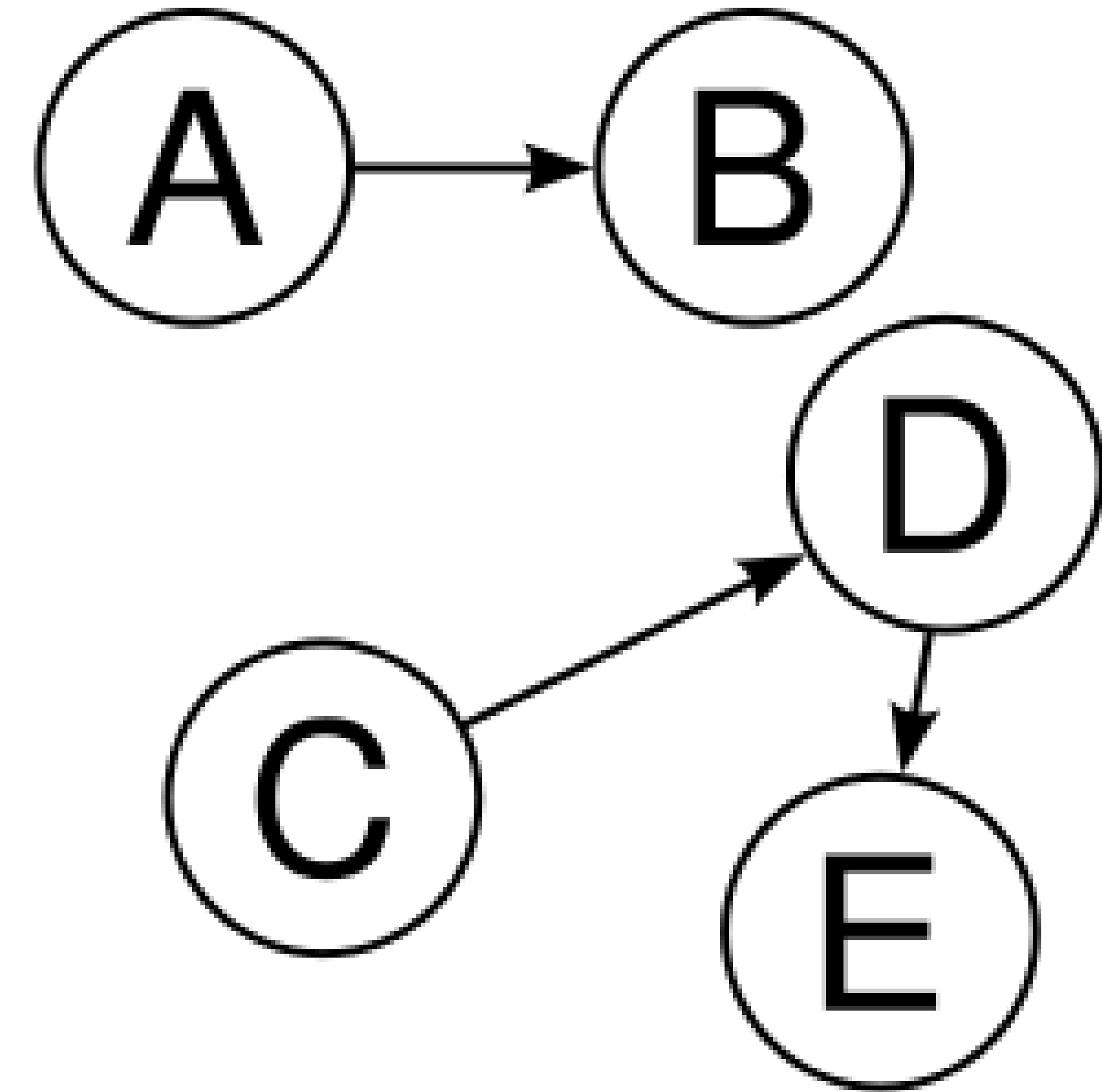
Stop the world

- Все потоки останавливаются
 - Проходим по всем узлам графа, помечая достижимые узлы
 - ????
 - Profit



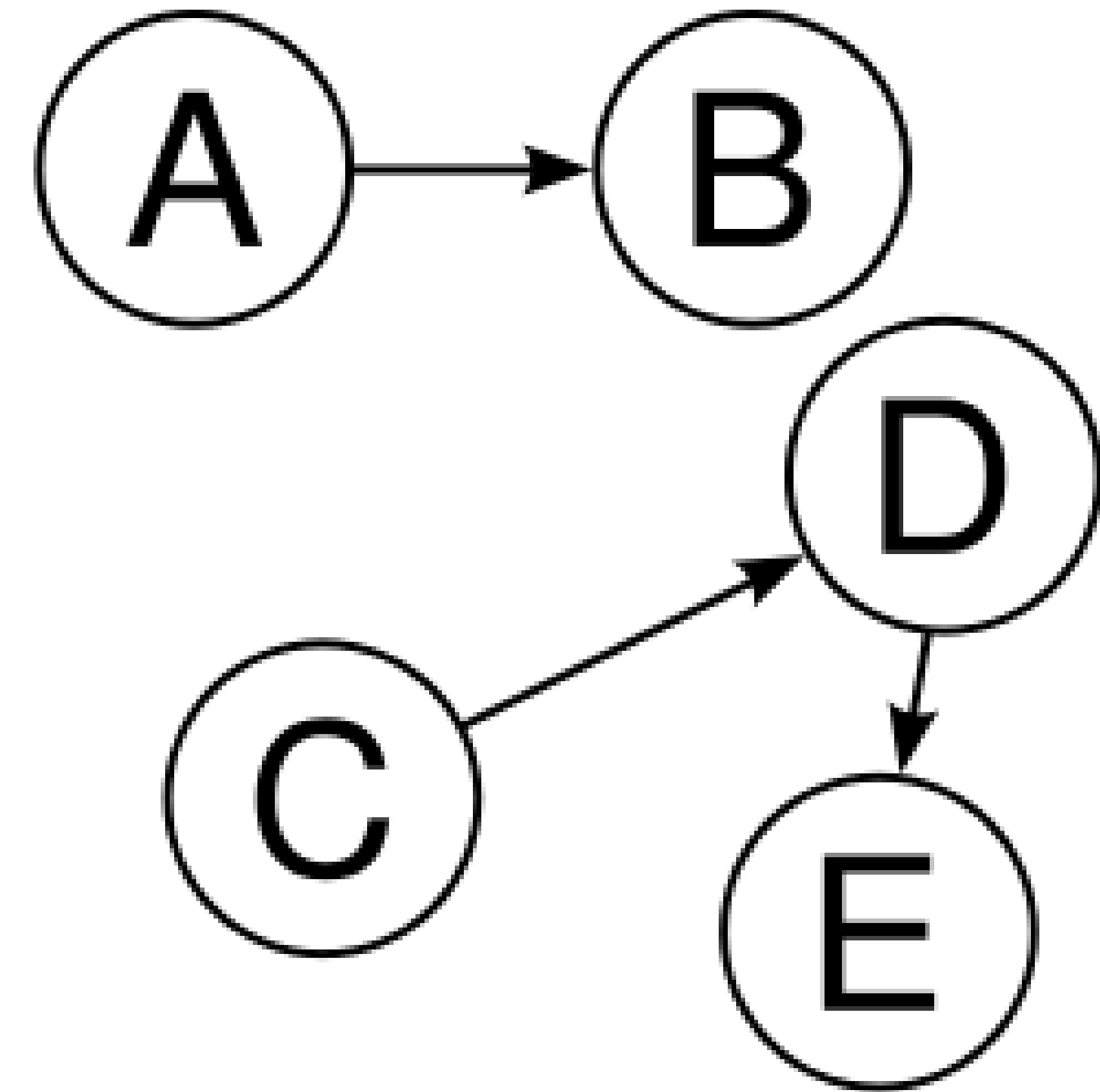
Stop the world

- Все потоки останавливаются
 - Проходим по всем узлам графа, помечая достижимые узлы
 - ????
 - Profit



- На современном железе больше 1 ядра!

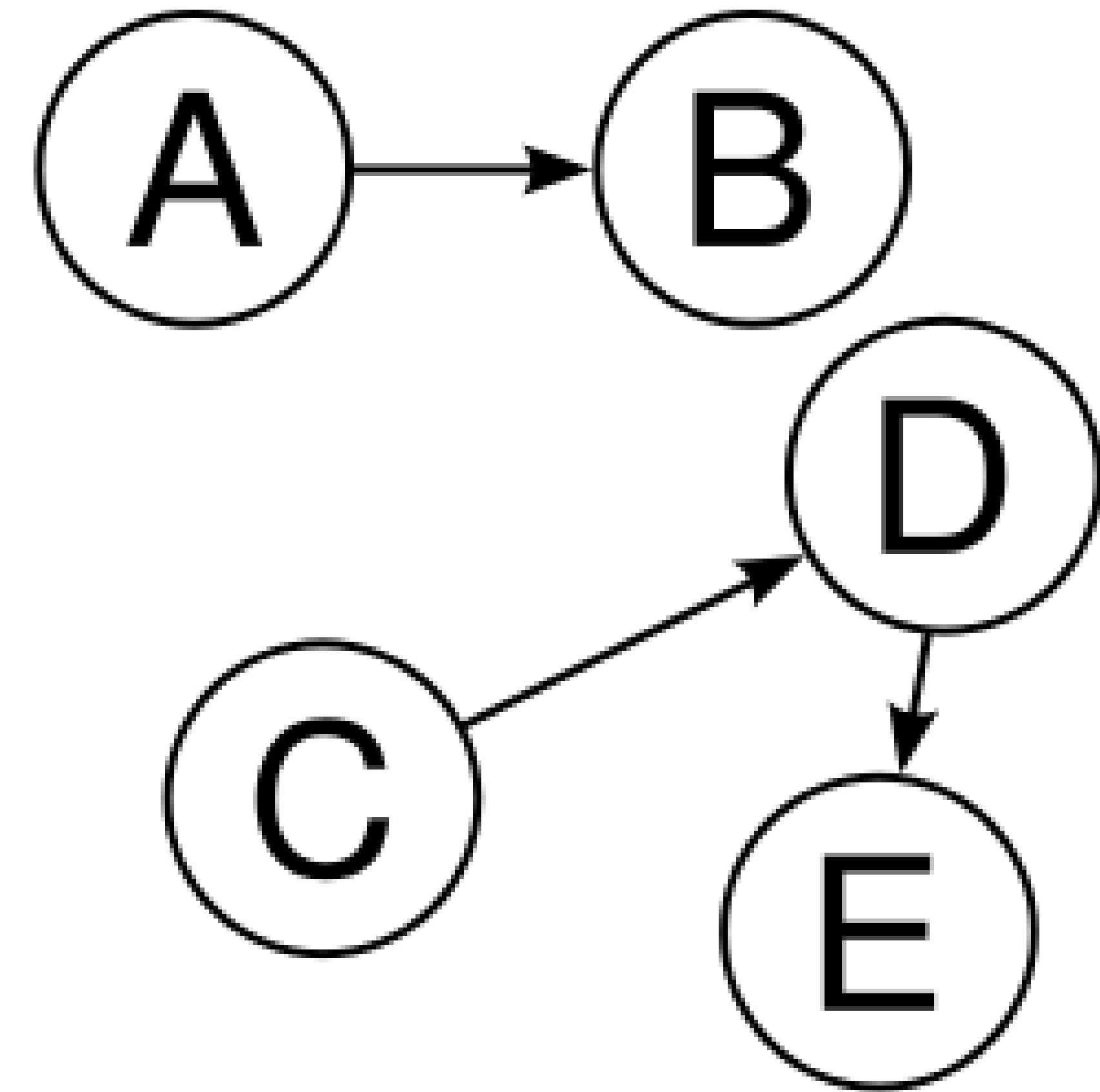
Stop the world



- Все потоки останавливаются
 - Проходим по всем узлам графа, помечая достижимые узлы
 - ????
 - Profit
- На современном железе больше 1 ядра!
- При каждой сборке мусора мы проходимся по всем узлам → постоянно перепроверяя живые узлы

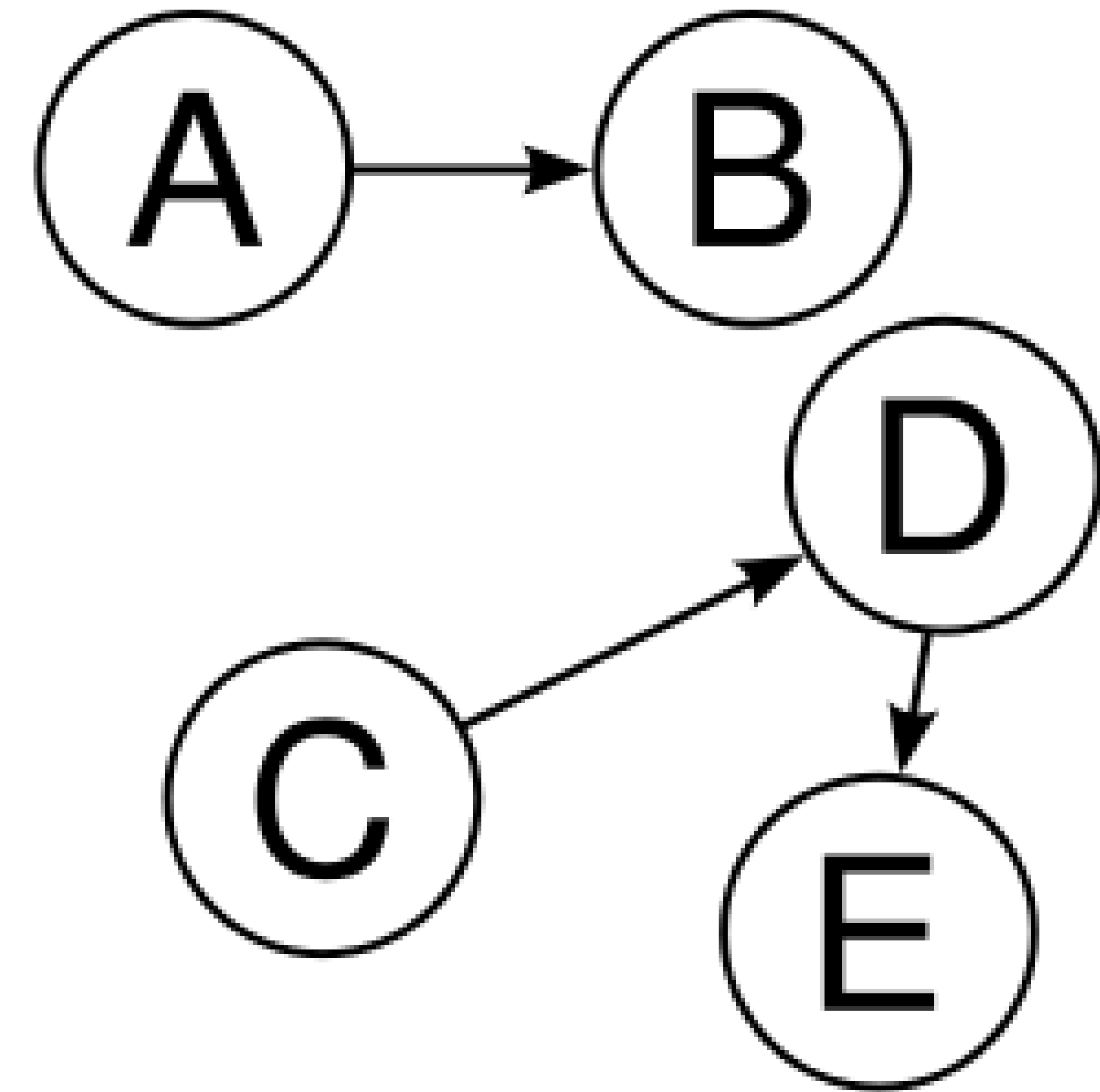
No stop the world

- Ничего не останавливается



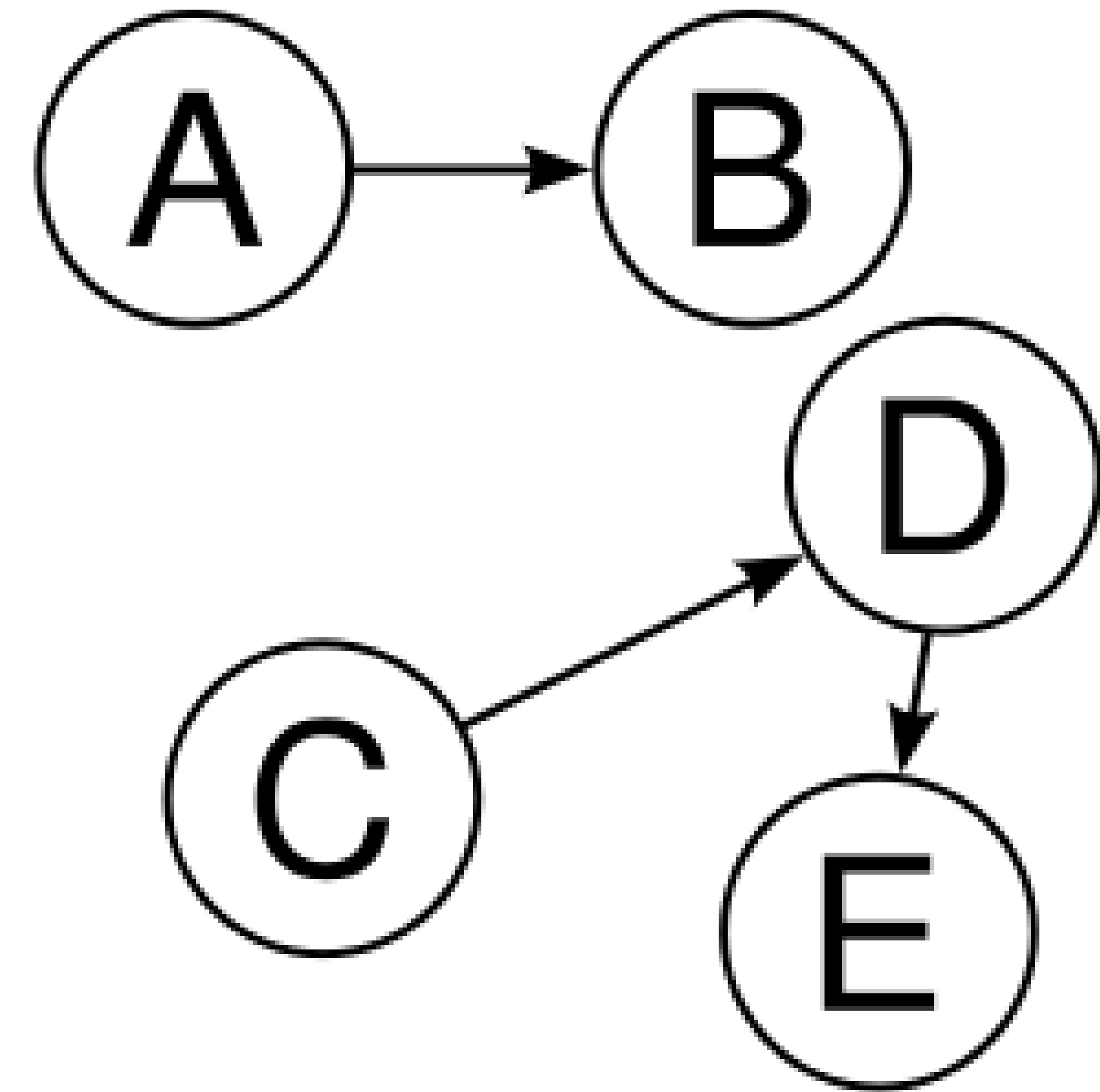
No stop the world

- Ничего не останавливается
 - Многопоточно разбираем мусор



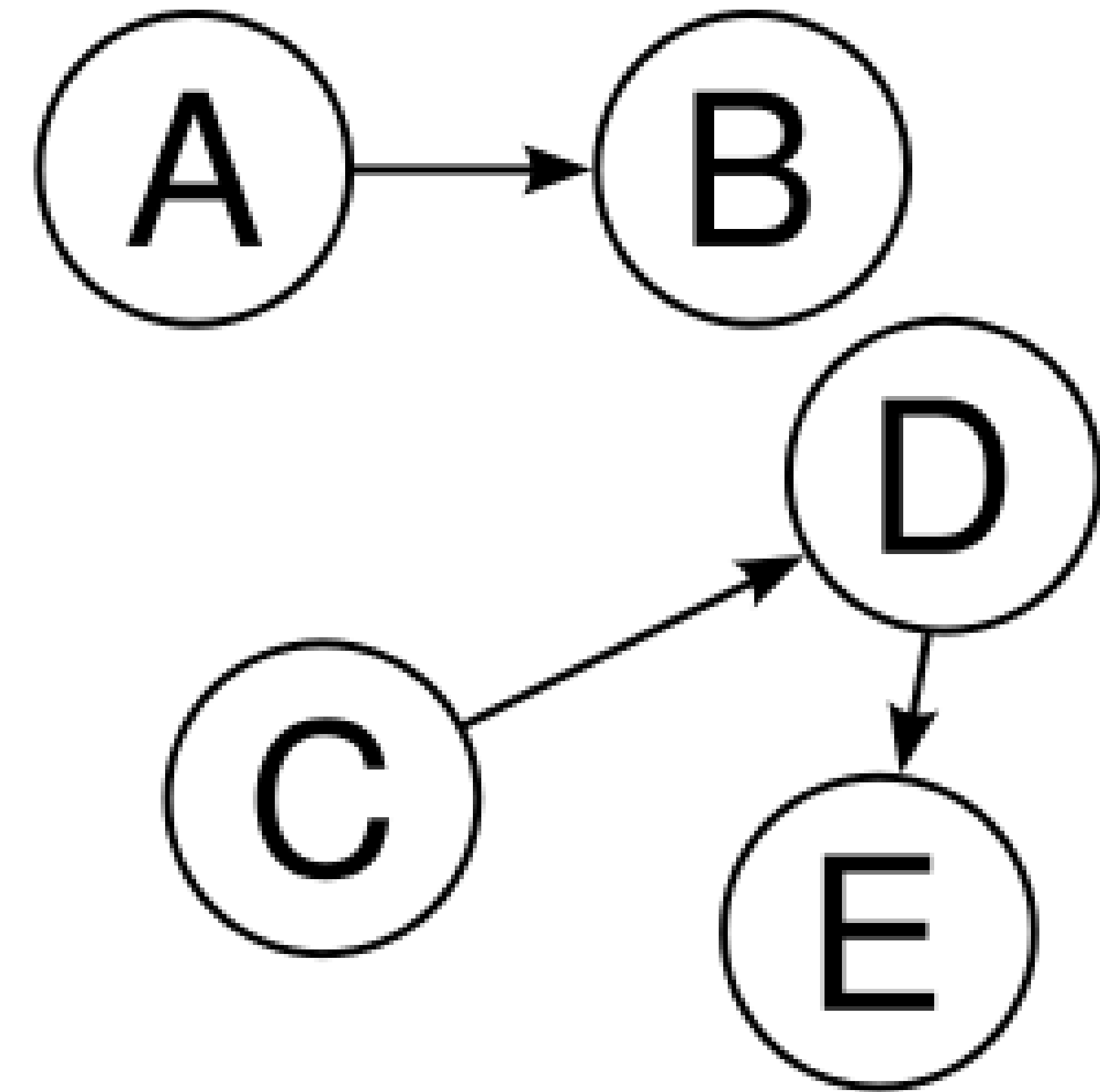
No stop the world

- Ничего не останавливается
 - Многопоточно разбираем мусор
 - Ура!



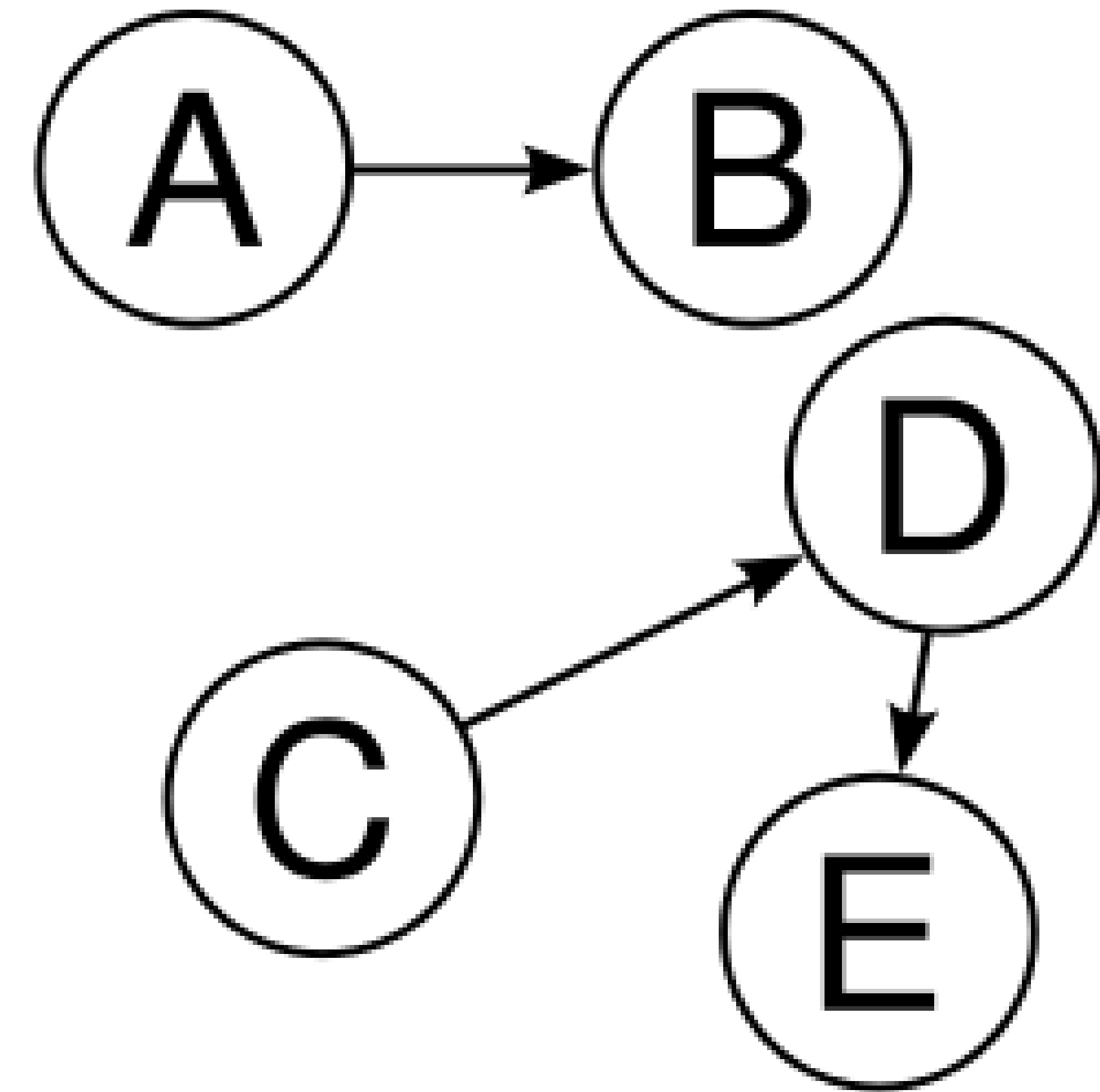
No stop the world

- Ничего не останавливается
 - Многопоточно разбираем мусор
 - Ура!
 - НО

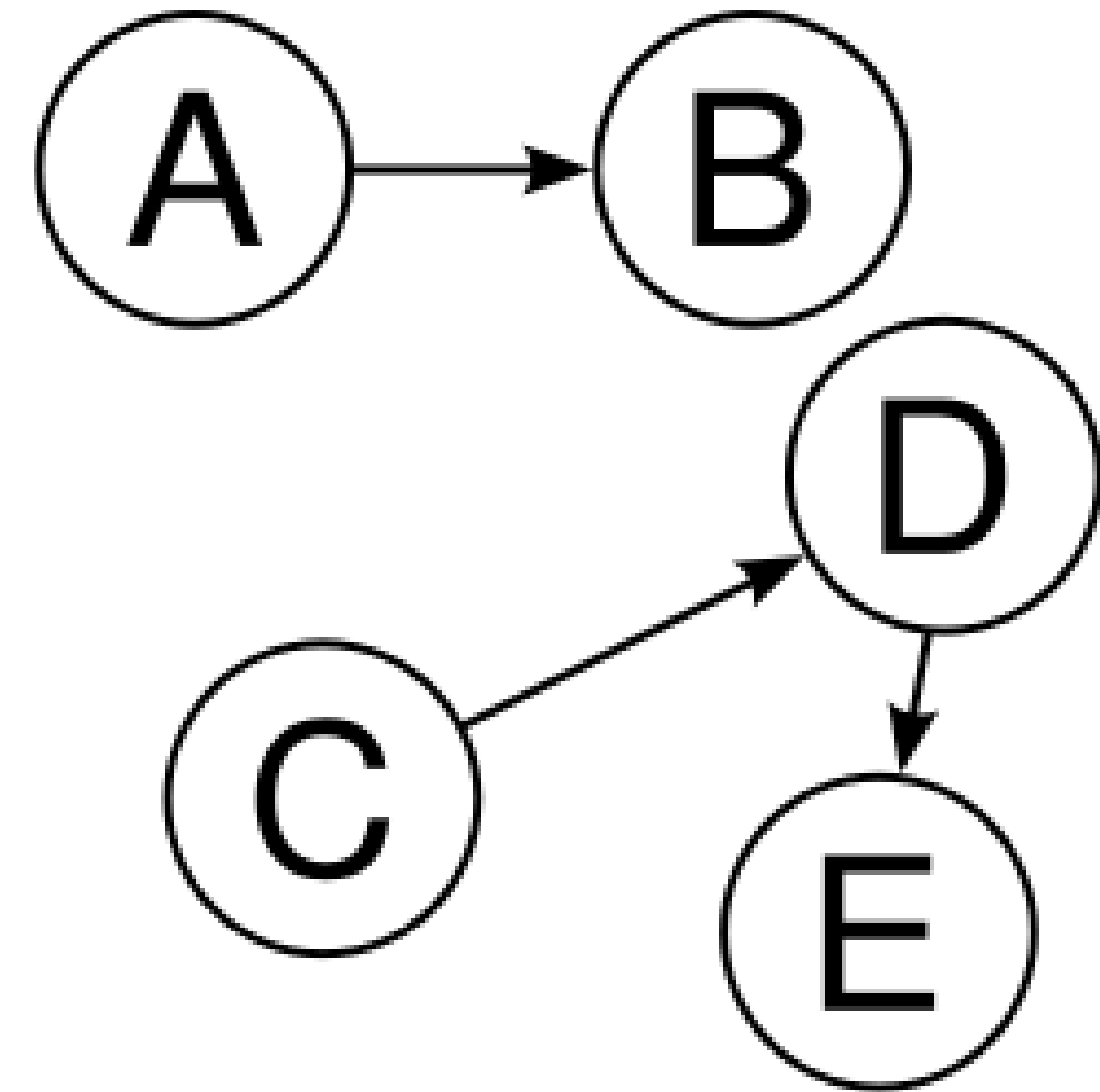


No stop the world

- Ничего не останавливается
 - Многопоточно разбираем мусор
 - Ура!
 - НО
 - Нам нужно синхронизировать все потоки через атомарные инструкции

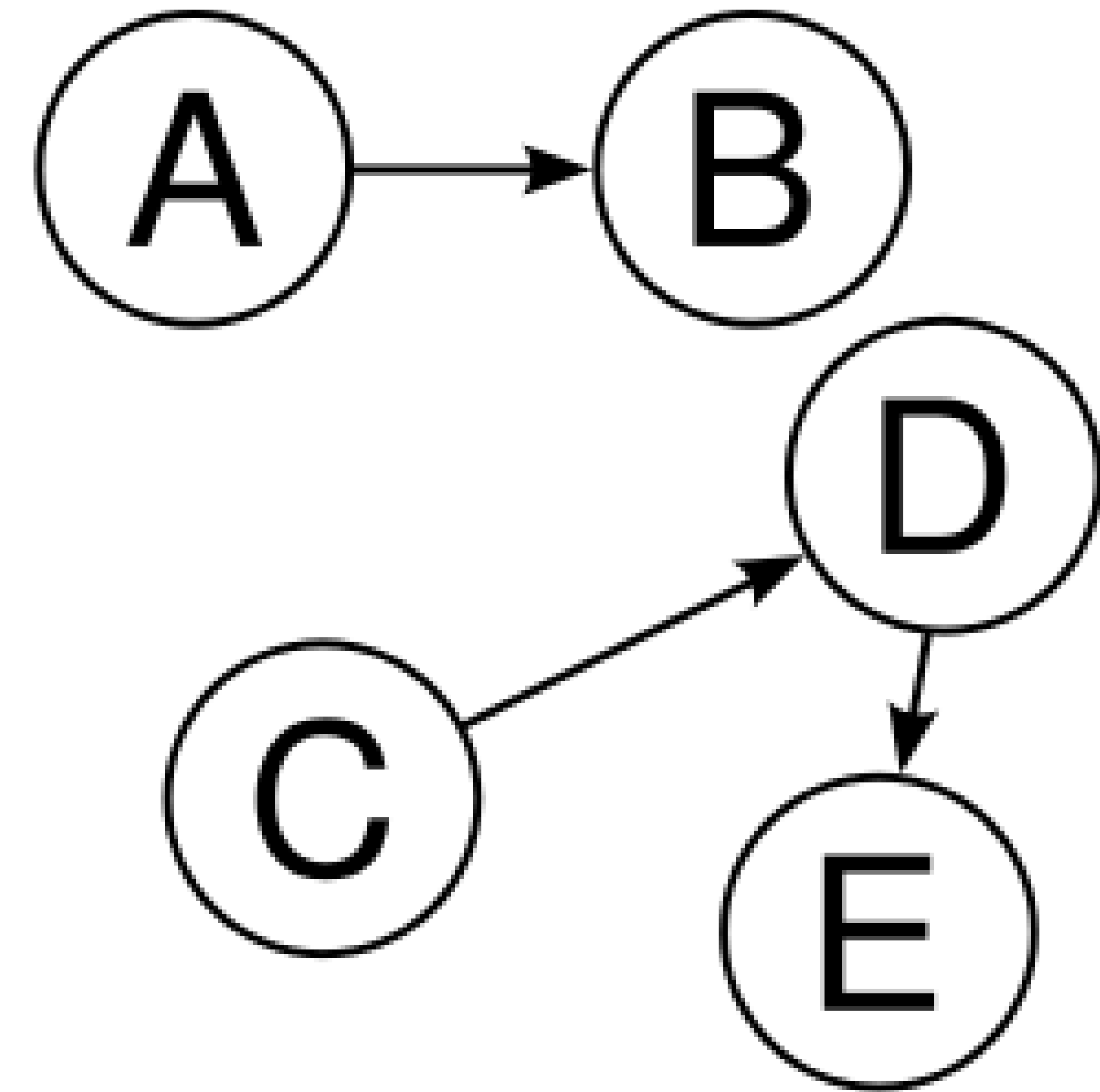


No stop the world



- Ничего не останавливается
 - Многопоточно разбираем мусор
 - Ура!
 - НО
 - Нам нужно синхронизировать все потоки через атомарные инструкции
 - Это очень дорого, если не мы последними трогали ресурс

No stop the world



- Ничего не останавливается
 - Многопоточно разбираем мусор
 - Ура!
 - НО
 - Нам нужно синхронизировать все потоки через атомарные инструкции
 - Это очень дорого, если не мы последними трогали ресурс
 - Мы делаем это при каждой сборке мусора!

**+ Не проходим
лишние разы**

structures

```
struct list_node {  
    shared_ptr<list_node> next;    // atomic refcounter  
    shared_ptr<list_node> prev;    // atomic refcounter  
};  
  
struct slist_node {  
    shared_ptr<slist_node> next;    // atomic refcounter  
};
```

Это ещё не всё!

```
shared_ptr<Object> some;
```


Это ещё не всё!

```
shared_ptr<Object> some;           // Скоро будет больно!
```

Это ещё не всё!

```
shared_ptr<Object> some;           // Скоро будет больно!
```

```
shared_ptr<Object> prev = some.a; // ...
```

Это ещё не всё!

```
shared_ptr<Object> some;           // Скоро будет больно!  
shared_ptr<Object> prev = some.a; // ...  
prev.a = some;                    // Циклические ссылки!
```

Trade-offs

Trade-offs

- Плавающий мусор

Trade-offs

- Плавающий мусор
 - ... или удар по производительности

Trade-offs

- Плавающий мусор
 - ... или удар по производительности
- Останавливать все потоки и тормозить с непредсказуемыми задержками

Trade-offs

- Плавающий мусор
 - ... или удар по производительности
- Останавливать все потоки и тормозить с непредсказуемыми задержками
 - ... или ничего не останавливать и тратить CPU такты впустую

Trade-offs

- Плавающий мусор
 - ... или удар по производительности
- Останавливать все потоки и тормозить с непредсказуемыми задержками
 - ... или ничего не останавливать и тратить CPU такты впустую
- Плата за Object*

C++ vs. (Java + C#)

C# ?? Java

C# ?? Java

- См. «Сборщики Мусора»

Logstash

– Программа для сбора, трансформации и складирования логов.

Бесплатное и очень популярное Open Source приложение на Java.

Что может пойти не так?

Logstash

– Программа для сбора, трансформации и складирования логов.

Бесплатное и очень популярное Open Source приложение на Java.

Что может пойти не так?

| %CPU | %MEM | COMMAND |
|-------|------|---------|
| 505,5 | 2,9 | java |
| 66,9 | 1,4 | daemon |

Слабые места C++

C++

C++

- скрытый высокий порог вхождения
 - неограниченные возможности
 - проблемы с безопасностью

C++

- скрытый высокий порог вхождения
 - неограниченные возможности
 - проблемы с безопасностью
- отсутствие изкоробочности
 - крошечная стандартная библиотека
 - отсутствие готовой инфраструктуры

ОК, а что делать то?

Итоги

Итоги

Итоги

- C++ везде

Итоги

- C++ везде
- C++ крайне популярен

Итоги

- C++ везде
- C++ крайне популярен
- C++ - самый производительный инструмент для написания больших приложений

Итоги

- C++ везде
- C++ крайне популярен
- C++ - самый производительный инструмент для написания больших приложений
- Относитесь со скепсисом к бенчмаркам

Итоги

- C++ везде
- C++ крайне популярен
- C++ - самый производительный инструмент для написания больших приложений
- Относитесь со скепсисом к бенчмаркам
- Не верьте managed языкам, которые говорят что они быстрее C++

Итоги

- C++ везде
- C++ крайне популярен
- C++ - самый производительный инструмент для написания больших приложений
- Относитесь со скепсисом к бенчмаркам
- Не верьте managed языкам, которые говорят что они быстрее C++
- C++ не идеален!...

Итоги

- C++ везде
- C++ крайне популярен
- C++ - самый производительный инструмент для написания больших приложений
- Относитесь со скепсисом к бенчмаркам
- Не верьте managed языкам, которые говорят что они быстрее C++
- C++ не идеален!...
 - ...но это исправимо

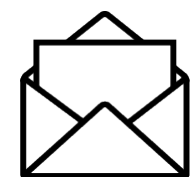
Спасибо

Полухин Антон

Старший разработчик Yandex.Taxi



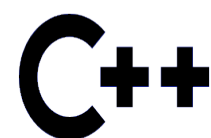
antoshkka@gmail.com



antoshkka@yandex-team.ru



<https://github.com/apolukhin>



РГ21 C++ РОССИЯ

<https://stdcpp.ru/>

