

# Zmienne warunku

CODERS SCHOOL

<https://coders.school>



Łukasz Ziobroń  
lukasz@coders.school

# Łukasz Ziobroń & Bartosz Szurgot - autorzy



# Łukasz Ziobroń

Not only programming experience:

- C++ and Python developer @ Nokia & Credit Suisse
- Scrum Master @ Nokia & Credit Suisse
- Code Reviewer @ Nokia
- Webmaster (HTML, PHP, CSS) @ StarCraft Area

Training experience:

- C++ trainings @ Coders School
- Practical Aspects Of Software Engineering @ PWr, UWr
- Nokia Academy @ Nokia
- Internal corporate trainings

Public speaking experience:

- Academic Championships in Team Programming
- code::dive conference
- code::dive community



# Zmienne warunku - Agenda

- Zadanie 1: kolejka FIFO
- Problem – grzanie CPU
- Aktywne czekanie
- Zmienna warunku (condition variable)
- Zadanie 2: kolejka FIFO przyjazna Arktyce
- Zmienna warunku – szczegóły
- Zadanie domowe: ping-pong

# Zadanie 1: kolejka FIFO

```
template <typename T>
class ThreadsafeQueue {
    deque<T> queue_;
    // TODO: Make it thread-safe :)

public:
    void push(const T & element) {
        queue_.push_front(element);
    }
    T pop() {
        auto top = queue_.back();
        queue_.pop_back();
        return top;
    }
    bool empty() const {
        return queue_.empty();
    }
};
```

- W pliku 01\_threadsafe\_queue.cpp znajduje się wybrakowana implementacja kolejki FIFO
- Napraw wątek textProducer, aby generował prawidłowe napisy:
  - This is random text number 0
  - This is random text number 1
  - ...
  - This is random text number n
- Zabezpiecz operacje na kolejce przed dostępem z wielu wątków (make it thread-safe 😊)
- Jaki problem widzisz?

# Zadanie 1 – rozwiązanie: naprawione wyświetlanie tekstu

```
void produceText(StringQueue & sq, int number) {  
    for (int i = 0; i < number; i++)  
        sq.push("This is random text number "  
                + i);  
}
```

```
void produceText(StringQueue & sq, int number) {  
    for (int i = 0; i < number; i++)  
        sq.push("This is random text number " +  
                to_string(i));  
}
```

- W pliku 01\_threadsare\_queue.cpp znajduje się wybrakowana implementacja kolejki FIFO
- Napraw wątek textProducer, aby generował prawidłowe napisy:
  - This is random text number 0
  - This is random text number 1
  - ...
  - This is random text number n
- Zabezpiecz operacje na kolejce przed dostępem z wielu wątków (make it thread-safe 😊)
- Jaki problem widzisz?

# Zadanie 1 – rozwiązanie: threadsafe queue

```
template <typename T>
class ThreadsafeQueue {
    deque<T> queue_;
    mutable mutex m_;
    using Lock = lock_guard<mutex>;

public:
    void push(const T & element) {
        Lock l(m_);
        queue_.push_front(element);
    }
    T pop() {
        Lock l(m_);
        auto top = queue_.back();
        queue_.pop_back();
        return top;
    }
    bool empty() const {
        Lock l(m_);
        return queue_.empty();
    }
};
```

- W pliku 01\_threadsare\_queue.cpp znajduje się wybrakowana implementacja kolejki FIFO
- Napraw wątek textProducer, aby generował prawidłowe napisy:
  - This is random text number 0
  - This is random text number 1
  - ...
  - This is random text number n
- Zabezpiecz operacje na kolejce przed dostępem z wielu wątków (make it thread-safe 😊)
- Jaki problem widzisz?



Problem – grzanie CPU i efekt cieplarniany





# Aktywne czekanie

```
void saveToFile(StringQueue & sq) {  
    ofstream file("/tmp/sth.txt");  
    while (file) {  
        while (sq.empty()) { /* nop */ }  
        file << sq.pop() << endl;  
    }  
}
```

- Aktywne czekanie (busy waiting) to stan, w którym wątek ciągle sprawdza, czy został spełniony pewien warunek
- Inna nazwa tego problemu to wirująca blokada (spinlock)
- Problem rozwiązuje zmienna warunku (condition variable)

```
1  [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||98.0%]  
2  [|||||||6.6%]  
Mem[|||||||||||||||||||||1.02G/4.86G]  
Swp[|||||||||||||||||0K/1.39G]  
Tasks: 103, 254 thr; 2 running  
Load average: 0.51 0.20 0.20  
Uptime: 00:25:13
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
3270	ziobron	20	0	167M	1988	1832	R	100.	0.0	0:32.97	./a.out

# Zmienna warunku (condition variable)

- `#include <condition_variable>`
- `std::condition_variable`
- Najważniejsze operacje
  - `wait()` - oczekuje na zmianę - blokuje obecny wątek dopóki nie zostanie on wybudzony
  - `notify_one()` - wybudza jeden z wątków oczekujących na zmianę. Nie mamy kontroli nad tym, który z wątków zostanie powiadomiony.
  - `notify_all()` - wybudza wszystkie wątki czekające na zmianę. Wątki te mogą konkurować o zasoby.

## Zadanie 2: kolejka FIFO przyjazna Arktyce

```
template <typename T>
class WaitQueue {
    deque<T> queue_;
    mutable mutex m_;
    using Lock = lock_guard<mutex>;

public:
    void push(const T & element) {
        Lock l(m_);
        queue_.push_front(element);
    }
    T pop() {
        Lock l(m_);
        auto top = queue_.back();
        queue_.pop_back();
        return top;
    }
    bool empty() const {
        Lock l(m_);
        return queue_.empty();
    }
};
```

- Popraw kod z pliku 02\_wait\_queue.cpp tak, aby używał zmiennej warunkowej zamiast aktywnego czekania

# Zadanie 2 - rozwiązanie

```
// includes
```

```
template <typename T>
class WaitQueue {
    deque<T> queue_;
    mutable mutex m_;
    condition_variable nonEmpty_;
    using Lock = unique_lock<mutex>;

public:
    void push(const T & element) {
        Lock l(m_);
        queue_.push_front(element);
        nonEmpty_.notify_all();
    }
    T pop() {
        Lock l(m_);
        auto hasData = [&]{ return not queue_.empty(); };
        nonEmpty_.wait(l, hasData);
        auto top = queue_.back();
        queue_.pop_back();
        return top;
    }
};
```

```
using StringQueue = WaitQueue<string>;
```

```
void provideData(StringQueue & sq) {
    string txt;
    while (getline(cin, txt))
        sq.push(txt);
}
```

```
void saveToFile(StringQueue & sq) {
    ofstream file("/tmp/sth.txt");
    while (file)
        file << sq.pop() << endl;
}
```

```
void produceText(StringQueue & sq, int number) {
    for (int i = 0; i < number; i++)
        sq.push("This is random text number " +
to_string(i));
}
```

```
int main() {
    // without changes
}
```

Zmienne warunki uszczęśliwiają foczki 😊





# Zmienna warunku - szczegóły

- `std::condition_variable` działa tylko z wyłącznymi blokadami (`unique_lock`)
- `std::condition_variable_any` działa z każdym rodzajem blokad (`shared_lock`)
- Są niekopiowalne
- Metoda `wait()` przyjmuje blokadę oraz opcjonalnie predykat, dzięki któremu zostaną wybudzone tylko te wątki, dla których warunek jest spełniony
- Wszystkie wątki, które czekają na zmiennej warunku muszą mieć zablokowany ten sam mutex. W przeciwnym wypadku jest niezdefiniowane zachowanie.
- Metody `wait_for()` i `wait_until()` przyjmują jeszcze odpowiednio punkt w czasie lub okres czasu do którego wątki będą czekać na wybudzenie. Po upływie tego czasu wątki zostaną wybudzone.
- Jeśli na zmiennej warunku czeka kilka wątków i każdy ma inny predykat, to użycie `notify_one()` może spowodować zakleszczenie. Wybudzony może zostać wątek, dla którego warunek nie został spełniony i jeśli żaden inny wątek nie zwoła `notify_one()` lub `notify_all()` to wszystkie będą czekać.

# Zadanie domowe: ping-pong

- 1 wątek wypisuje "ping" oraz kolejny numer
- 2 wątek wypisuje "pong" oraz kolejny numer
- Zaczyna wątek ping, a kończy zawsze pong. Wątki muszą pracować na przemian. Nie mogą być 2 pingi lub pongi po sobie. Program nie może zakończyć się pingiem, na który nie będzie odpowiedzi – ponga.
- Zakończenie działania programu ma nastąpić albo po wykonanej liczbie odbić albo po limicie czasowym, w zależności które wystąpi pierwsze. Powód zakończenia powinien zostać wyświetlony na ekranie
- Parametry programu:
  - liczba odbić
  - limit czasowy (w sekundach)

```
$> g++ 03_ping_pong.cpp -lpthread  
-std=c++17 -fsanitize=thread  
$> ./a.out 1 10  
ping 0  
pong 0  
Ping reached repetitions limit  
Pong reached repetitions limit  
$> ./a.out 12 1  
ping 0  
pong 0  
ping 1  
pong 1  
ping 2  
pong 2  
Timeout
```

# Wskazówki

- Jeśli utkniesz:
  - Potrzebujesz mutexu i zmiennej warunkowej w klasie PingPong
  - Czekaj na zmiennej warunku za pomocą **wait\_for()** w funkcji **stop()**
  - Sprawdzaj liczbę powtórzeń w wątkach ping i pong
  - Użyj dodatkowej zmiennej **bool**, która powie wszystkim wątkom, aby się zakończyły. gdy nastąpią wymagane warunki. Użyj tutaj typu **atomic<bool>** (o nim później 😊)
  - Wątki ping i pong powinny za pomocą **wait()** sprawdzać warunek, czy to ich kolej na działanie. Użyj dodatkowej zmiennej **bool**, która zostanie użyta w predykcji przekazanym do **wait()**.
  - Wątek pong powinien zakończyć program po osiągnięciu limitu odbić

# CODERS SCHOOL

<https://coders.school>

ASK A NINJA



Łukasz Ziobroń  
[lukasz@coders.school](mailto:lukasz@coders.school)