

Biblioteka std::thread w C++

Wyżej wymieniona biblioteka, jest jedną z najprostszych metod pracy wielowątkowej. Opera się na podejściu stwórz wątek wykonujący daną funkcję z danymi parametrami.

Link do dokumentacji:

<https://en.cppreference.com/w/cpp/thread/thread>

Jak widać biblioteka nie ma jakoś bardzo dużo elementów, dlatego jest bardzo przyjemna do nauki.

Uruchomienie wątku:

KOD1:

```
1  #include <cstdio>
2  #include <thread>
3  #include <windows.h>
4
5  int action(int id){
6      printf("Uruchamiam watek %d\n", id);
7      Sleep(10*1000); //10 sekund
8      printf("Koncze watek %d\n", id);
9      return 0;
10 }
11
12 int main(){
13     //tworzenie watku
14     std::thread t1(action, 1); //konstruktor klasy t1 przyjmuje minimum wskaźnik na funkcję do wykonania
15     std::thread t2(action, 2); //funkcja ta może coś zwracać, ale może zwracać też void
16     std::thread t3(action, 3); //dalsze parametry zostaną przekazane do podanej funkcji
17
18     t1.join(); //synchronizacja
19     t2.join(); //wątek główny ma tu poczekać na te 3 wątki
20     t3.join(); //inaczej program by się zakończył wcześniej bo wątki trwają minimum 10 sekund
21
22     printf("Koniec programu \r\n");
23
24     return 0;
25 }
```

Jak widać oprócz uruchomienia dokonano synchronizacji. Wątek główny poczeka na wątki utworzone. Same wątki nie robią nic ciekawego. Można sklasyfikować je jako IO bound.

Zadania do KOD1:

1. Uruchom program wiele razy i porównaj wyjście.
2. Usuń wszystkie 3 joiny, skompiluj i sprawdź wyjście.
3. Usuń tylko jeden join, skompiluj i sprawdź co zostanie wyświetlone (najlepiej więcej niż raz).
4. Zmodyfikuj program tak by wypisywane było id uzyskane z funkcji get\_id()  
[https://en.cppreference.com/w/cpp/thread/get\\_id](https://en.cppreference.com/w/cpp/thread/get_id)
5. Zmodyfikuj funkcję action w taki sposób by czas uśpienia dostawał jako parametr, następnie przetestuj zmiany. Sprawdź:
  - w jakiej kolejności zostaną zamknięte wątki względem podanych czasów

Otwieranie bardzo dużej ilości wątków:

Poniższy przykład pokazuje jak otworzyć i kontrolować większą ilość wątków np. pobraną od użytkownika.

KOD2:

```
1  #include <stdio>
2  #include <thread>
3  #include <windows.h>
4
5  void action(int id){
6      printf("Uruchamiam watek %d\n", id);
7      Sleep(5*1000); //5 sekund
8      printf("Koncze watek %d\n", id);
9  }
10
11 int main(){
12     int thread_count = 8;
13
14     //alokacja tablicy, która będzie przechowywać wskaźniki na wątki
15     std::thread** threads = new std::thread*[thread_count];
16
17     //otwieranie wątków
18     for(int i = 0; i < thread_count; i++){
19         threads[i] = new std::thread(action, i); //wykorzystuje i jako id danego wątku
20     }
21
22     //wątki pracują, ale trzeba je zsynchronizować
23     for(int i = 0; i < thread_count; i++){
24         threads[i]->join();
25     }
26
27     //alokowaliśmy pamięć więc pasuje ją zwolnić
28     for(int i = 0; i < thread_count; i++){
29         delete threads[i];
30     }
31     delete[] threads;
32
33     printf("Koniec programu \r\n");
34
35     return 0;
36 }
```

Zadania do KOD2:

1. Uruchom program wiele razy i porównaj wyjścia.
2. Zmodyfikuj program tak by ilość wątków była pobierana od użytkownika.
3. Zmodyfikuj program tak by zamiast tablicy przechowującej wskaźniki na wątki użyć wektora przechowującego wskaźniki na wątki.