Vlákna v C++

FILIP ŠIMOVIČ 5ZY037

Na zopakovanie

- Proces
  - Beh programu
- Vlákno
  - Proces ich obsahuje jedno alebo viac
- Deadlock
  - „uviaznutie“ – procesy na seba navzájom čakajú
- Atomická operácia
  - Nesmie byť počas čítania prerušená

Motivácia

- Pomalé reakcie
- Multitasking
- Spolupráca s používatelom
- Zrýchlenie programu

Threading Library

Unquestionably, the most important addition to C++11 from a programmer's perspective is concurrency(!=konkurencia, ale paralelizmus).

C++11 has a thread class that represents an execution thread, promises and futures, which are objects that are used for synchronization in a concurrent environment, the async()function template for launching concurrent tasks, and the thread_local storage type for declaring thread-unique data.

Podpora vlákien v C++

Štandardná knižnica poskytuje pre prácu s vláknami viacero prostriedkov:

- std::thread
- std::mutex, std::shared_mutex
- std::condition_variable
- std::atomic

Vlákna v C++

```
1   #include <iostream>
2   #include <thread>
3
4   //This function will be called from a thread
5
6   void call_from_thread() {
7       std::cout << "Hello, World" << std::endl;
8   }
9
10  int main() {
11      //Launch a thread
12      std::thread t1(call_from_thread);
13
14      //Join the thread with the main thread
15      t1.join();
16
17      return 0;
18  }
```

```
CppApplication_1 (Build, Run) - simovic@frios2.fri.uniza.sk:22 ×
Hello, World
```

Mutual exclusion

Mutex je objekt, ktorý môže byť v dvoch stavoch:

- *zamknutý* (*locked*)
- *odomknutý* (*unlocked*)

A **thread** is : Each person
The **mutex** is : The door handle
The **lock** is : The person's hand
The **resource** is : The phone

```cpp
#include <iostream>
#include <thread>
#include <mutex>

std::mutex m;//you can use std::lock_guard if you want to be exception safe
int i = 0;

void makeACallFromPhoneBooth()
{
    m.lock();//man gets a hold of the phone booth door and locks it. The other men wait out
      //man happily talks to his wife from now....
      std::cout << i << " Hello Wife" << std::endl;
      i++;//no other thread can access variable i until m.unlock() is called
      //...until now, with no interruption from other men
    m.unlock();//man lets go of the door handle and unlocks the door
}

int main()
{
    //This is the main crowd of people uninterested in making a phone call

    //man1 leaves the crowd to go to the phone booth
    std::thread man1(makeACallFromPhoneBooth);
    //Although man2 appears to start second, there's a good chance he might
    //reach the phone booth before man1
    std::thread man2(makeACallFromPhoneBooth);
    //And hey, man3 also joined the race to the booth
    std::thread man3(makeACallFromPhoneBooth);

    man1.join();//man1 finished his phone call and joins the crowd
    man2.join();//man2 finished his phone call and joins the crowd
    man3.join();//man3 finished his phone call and joins the crowd
    return 0;
}
```

```cpp
#include <mutex>

std::mutex m;//you can use std::lock_guard if you want to be exception safe
int i = 0;

void makeACallFromPhoneBooth()
{
    m.lock();//man gets a hold of the phone booth door and locks it. The other men wait outs
      //man happily talks to his wife from now....
      std::cout << i << " Hello Wife" << std::endl;
      i++;//no other thread can access variable i until m.unlock() is called
      //...until now, with no interruption from other men
    m.unlock();//man lets go of the door handle and unlocks the door
}

int main()
{
    //This is the main crowd of people uninterested in making a phone call

    //man1 leaves the crowd to go to the phone booth
    std::thread man1(makeACallFromPhoneBooth);
    //Although man2 appears to start second, there's a good chance he might
    //reach the phone booth before man1
    std::thread man2(makeACallFromPhoneBooth);
    //And hey, man3 also joined the race to the booth
    std::thread man3(makeACallFromPhoneBooth);

    man1.join();//man1 finished his phone call and joins the crowd
    man2.join();//man2 finished his phone call and joins the crowd
    man3.join();//man3 finished his phone call and joins the crowd
    return 0;
}
```

Výstup:

```
0 Hello Wife
1 Hello Wife
2 Hello Wife
```

Condition Variables

- Na rozdiel od mutexov slúžia na komunikáciu medzi vláknami – synchonizáciu
- Vlákna
  - čakajú na signál od iného vlákna
  - môžu sa prebudiť samovoľne
  - nemusia byť schopné v súčasnom stave pokračovať

Notification	
`notify_one`	notifies one waiting thread (public member function)
`notify_all`	notifies all waiting threads (public member function)
Waiting	
`wait`	blocks the current thread until the condition variable is woken up (public member function)
`wait_for`	blocks the current thread until the condition variable is woken up or after the specified timeout duration (public member function)
`wait_until`	blocks the current thread until the condition variable is woken up or until specified time point has been reached (public member function)

Notification

`notify_one`	notifies one waiting thread (public member function)
`notify_all`	notifies all waiting threads (public member function)

Waiting

`wait`	blocks the current thread until the condition variable is woken up (public member function)
`wait_for`	blocks the current thread until the condition variable is woken up or after the specified timeout duration (public member function)
`wait_until`	blocks the current thread until the condition variable is woken up or until specified time point has been reached (public member function)

Atomické premenné

Alternatíva k uzamknutiu jedenej premennej.

Menej nákladná operácia ako mutex.

Pre synchronizáciu viac premenných treba aj tak použiť mutex.

General atomic operations

is_lock_free	Is lock-free (public member function)
store	Modify contained value (public member function)
load	Read contained value (public member function)

Zdroje

- http://www.cplusplus.com/reference/thread/thread/
- https://www.sallyx.org/sally/c/linux/threads
- https://cw.fel.cvut.cz/wiki/
- https://solarianprogrammer.com/2011/12/16/cpp-11-thread-tutorial/
- https://stackoverflow.com/questions/266168/simple-example-of-threading-in-c
- https://frios2.fri.uniza.sk/~chochlik/frios/frios/sk.html
- https://smartbear.com/blog/develop/the-biggest-changes-in-c11-and-why-you-should-care/

Ďakujem za pozornosť!