|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет  имени Н. Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н. Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика, искусственный интеллект и системы управления» (ИУ) |

|  |  |
| --- | --- |
| КАФЕДРА | «Информационная безопасность» (ИУ8) |

Лабораторная работа № 7

ПО КУРСУ

«Алгоритмические языки»

# на тему «Изучение потоковой многозадачности»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-24 |  |  |  | Соболева Д.Е. |
|  | (Группа) |  |  |  | (И. О. Фамилия) |
|  |  |  |  |  |  |
| Преподаватель: |  |  |  |  | Барыкин Д. В |
|  |  |  |  |  | (И.О. Фамилия) |

**Цель работы:**

Изучение и понимание концепции потоковой многозадачности с целью освоения эффективного управления параллельными процессами в программировании.

**Текст программы:**

1. **Future и async**

#include <iostream>

#include <thread>

#include <future>

#include <random>

void thread\_function(int n, const char\* name, double\* arr1, double\* arr2) {

std::default\_random\_engine generator;

std::uniform\_real\_distribution<double> distribution(1.0, 10.0);

for (int i = 0; i < n; ++i) {

std::cout << name << ":" << std::endl;

double num1 = arr1[i];

double num2 = arr2[i];

std::cout << "Operand 1: " << num1 << std::endl;

std::cout << "Operand 2: " << num2 << std::endl;

double result = num1 \* num2;

std::cout << "Result: " << result << std::endl;

std::cout << std::endl;

}

}

int main() {

const int n1 = 2;

const int n2 = 2;

const int n3 = 2;

double arr1[n1];

double arr2[n1];

std::default\_random\_engine generator;

std::uniform\_real\_distribution<double> distribution(1.0, 10.0);

for (int i = 0; i < n1; ++i) {

arr1[i] = distribution(generator);

arr2[i] = distribution(generator);

}

std::future<void> future1 = std::async(std::launch::async, thread\_function, n1, "thread1", arr1, arr2);

std::future<void> future2 = std::async(std::launch::async, thread\_function, n2, "thread2", arr1, arr2);

std::future<void> future3 = std::async(std::launch::async, [](int n) {

std::default\_random\_engine generator;

std::uniform\_real\_distribution<double> distribution(1.0, 10.0);

for (int i = 0; i < n; ++i) {

std::cout << "thread3:" << std::endl;

double random\_value = distribution(generator);

std::cout << "Random value: " << random\_value << std::endl;

std::cout << std::endl;

}

}, n3);

future1.get();

future2.get();

future3.get();

std::cout << "Main thread has finished." << std::endl;

return 0;

}

1. Thread
2. #include <iostream>
3. #include <thread>
4. #include <random>
5. #include <mutex>
6. using namespace std;
7. void thread\_function(int n, const char\* name, double\* arr1, double\* arr2) {
8. default\_random\_engine generator;
9. uniform\_real\_distribution<double> distribution(1.0, 10.0);
10. for (int i = 0; i < n; ++i) {
11. cout << name << ":" << std::endl;
12. double num1 = arr1[i];
13. double num2 = arr2[i];
14. cout << "Operand 1: " << num1 << std::endl;
15. cout << "Operand 2: " << num2 << std::endl;
16. double result = num1 + num2;
17. cout << "Result: " << result << std::endl;
18. cout << std::endl;
19. }
20. }
21. int main() {
22. const int n1 = 2;
23. const int n2 = 2;
24. const int n3 = 2;
25. double arr1[n1];
26. double arr2[n1];
27. default\_random\_engine generator;
28. uniform\_real\_distribution<double> distribution(1.0, 10.0);
29. for (int i = 0; i < n1; ++i) {
30. arr1[i] = distribution(generator);
31. arr2[i] = distribution(generator);
32. }
33. thread thread1(thread\_function, n1, "thread1", arr1, arr2);
34. thread thread2(thread\_function, n2, "thread2", arr1, arr2);
35. default\_random\_engine generator2;
36. uniform\_real\_distribution<double> distribution2(1.0, 10.0);
37. thread thread3([n3]() {
38. default\_random\_engine generator;
39. uniform\_real\_distribution<double> distribution(1.0, 10.0);
40. for (int i = 0; i < n3; ++i) {
41. cout << "thread3:" << std::endl;
42. double random\_value = distribution(generator);
43. cout << "Random value: " << random\_value << std::endl;
44. cout << std::endl;
45. }
46. });
47. thread1.join();
48. thread2.join();
49. thread3.join();
50. cout << "Main thread has finished." << std::endl;
51. return 0;
52. }

**Вопрос1 для 1:**

После запуска программы несколько раз: из-за использования ассинхронных задач, порядок выполнения потоков может меняться при каждом запуске программы, последовательность в которой в потоке начинает выполняться может быть различной. Результаты вывода значений от каждого потока могут отличаться при разных запусках программы.

**Вопрос2 для 1:**

После добавления задержки в 10мс между печатью имени потока и вывода значений, порядок выполнения потоков может немного измениться. Задержка помогает более явно увидеть, как каждый поток начинает выполнение.

**Вопрос1 для 2:**

При использовании данного класса и явного управления потоками, результаты так же могу меняться при каждом запуске программы. Порядок завершения потоков может быть разным из-за случайности генерации случайных чисел.

**Вопрос2 для 2:**

Добавление задержки в 10мс между печатью имени потока и вывода значений, влияет на порядок выполнения потоков.

**Вывод:**

После выполнения данной лабораторной работы, я изучила тему «Потоковая многозадачность» и увидела различия, которые могут происходить в программе.