



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 и системы управления»

КАФЕДРА \_\_\_\_\_ «Теоретическая информатика и компьютерные технологии»

**Лабораторная работа № 4**  
**по курсу «Распределение параллельных и распределённых**  
**программ»**  
**«Задача о пяти обедающих философах»**

Студент группы ИУ9-51Б Горбунов А. Д.

Преподаватель Царёв А. С.

*Москва 2024*

# 1 Задача

Суть задачи следующая. Пять философов сидят за круглым столом. Они проводят жизнь, чередуя приёмы пищи и размышления. В центра стола находится большое блюдо спагетти. Чтобы съесть порцию, каждому философу нужно две вилки. Однако, вилок всего пять: между каждой парой рядом сидящих философов лежат по одной вилке, и каждый философ может пользоваться только теми вилками, которые лежат рядом с ним, слева и справа. Философ не может брать две вилки одновременно: сначала он тратит некоторое время на то, чтобы взять одну, затем вторую. Однако, он может одновременно положить их на место.

Задача заключается в том, чтобы написать программу, моделирующую поведение философов. Очевидно, что раз вилок всего пять, то одновременно есть могут не более двух философов, и два сидящих рядом философа не могут есть одновременно. Для имитации периодов раздумий и приёмов пищи можно использовать генератор случайных чисел, позволяющий задавать времена их действий в определённом интервале. Имитация поведения каждого философа, по сути, разбивается на то, что в любой момент времени философ находится в одном из пяти состояний: размышляет, берёт левую вилку, берёт правую вилку, ест, кладёт вилки на место. Таким образом, вилки являются разделяемым ресурсом.

На программу накладываются условия:

1. Каждый философ, по сути, является потоком, и модель поведения у каждого из них должна быть одинаковой, кроме того, какие вилки они могут брать.

2. Накладывание блокировки по сути является действием по взятию вилки, поэтому накладывать блокировку сразу на обе вилки нельзя; последовательность действий должна быть «наложить блокировку – взять вилку – наложить вторую блокировку – взять вторую вилку».

3. Программа должна избегать ситуации взаимоблокировки: ситуации, в которой все философы голодны, то есть ни один из них не может взять себе две вилки (например, когда каждый держит по одной и не хочет её отдавать).

Запрограммировать остановку алгоритма по достижении контрольного времени (например, атомарной операцией над булевым флагом). В отчёте построить некоторый результат работы алгоритма, которая может быть в виде графика,

таблицы, лога или чего угодно ещё; главное условие состоит в том, чтобы по результатам можно было однозначно определить, чем в каждый момент времени был занят каждый философ (одно из пяти состояний).

Также рассмотреть вариант программы с увеличением количества философов до произвольного N.

## 2 Код решения

Файл main.cpp:

```
#include <iostream>
#include <thread>
#include <mutex>
#include <vector>
#include <chrono>
#include <condition_variable>

using namespace std;

const int NUM_PHILOSOPHERS = 5;
const int THINKING_TIME_MIN = 1;
const int THINKING_TIME_MAX = 3;
const int EATING_TIME_MIN = 1;
const int EATING_TIME_MAX = 3;
const int SIMULATION_TIME = 10;

mutex forks[NUM_PHILOSOPHERS];
condition_variable cv;
bool stop_simulation = false;

void philosopher(int id) {
    int left_fork = id;
    int right_fork = (id + 1) % NUM_PHILOSOPHERS;
```

```

while (!stop_simulation) {
    this_thread::sleep_for(chrono::seconds(THINKING_TIME_MIN + rand() %
    printf("Философ %d подумал.\n", id);

    unique_lock<mutex> lock_left(forks[left_fork]);
    printf("Философ %d взял левую вилку %d.\n", id, left_fork);

    unique_lock<mutex> lock_right(forks[right_fork], defer_lock);
    if (lock_right.try_lock()) {
        printf("Философ %d взял правую вилку %d.\n", id, right_fork);

        this_thread::sleep_for(chrono::seconds(EATING_TIME_MIN + rand() %
        printf("Философ %d поел.\n", id);

        printf("Философ %d отложил вилку %d и %d.\n", id, left_fork, right_fork);
    } else {

        printf("Философ %d не смог взять правую вилку %d.\n", id, right_fork);
    }
}

}

int main() {
    vector<thread> philosophers;

    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        philosophers.emplace_back(philosopher, i);
    }

    this_thread::sleep_for(chrono::seconds(SIMULATION_TIME));

    stop_simulation = true;
    cv.notify_all();
}

```

```
    for (auto& t : philosophers) {  
        t.join();  
    }  
  
    return 0;  
}
```

Лог работы программы:

```
goarty@GoComp:~/Documents/paral_program/lab_4/src$ c++ -o main.o main.cpp  
goarty@GoComp:~/Documents/paral_program/lab_4/src$ ./main.o
```

Философ 2 подумал.

Философ 2 взял левую вилку 2.

Философ 2 взял правую вилку 3.

Философ 0 подумал.

Философ 0 взял левую вилку 0.

Философ 0 взял правую вилку 1.

Философ 1 подумал.

Философ 3 подумал.

Философ 4 подумал.

Философ 4 взял левую вилку 4.

Философ 4 не смог взять правую вилку 0.

Философ 2 поел.

Философ 2 отложил вилку 2 и 3.

Философ 3 взял левую вилку 3.

Философ 3 взял правую вилку 4.

Философ 0 поел.

Философ 0 отложил вилку 0 и 1.

Философ 1 взял левую вилку 1.

Философ 1 взял правую вилку 2.

Философ 4 подумал.

Философ 2 подумал.

Философ 3 поел.

Философ 3 отложил вилку 3 и 4.

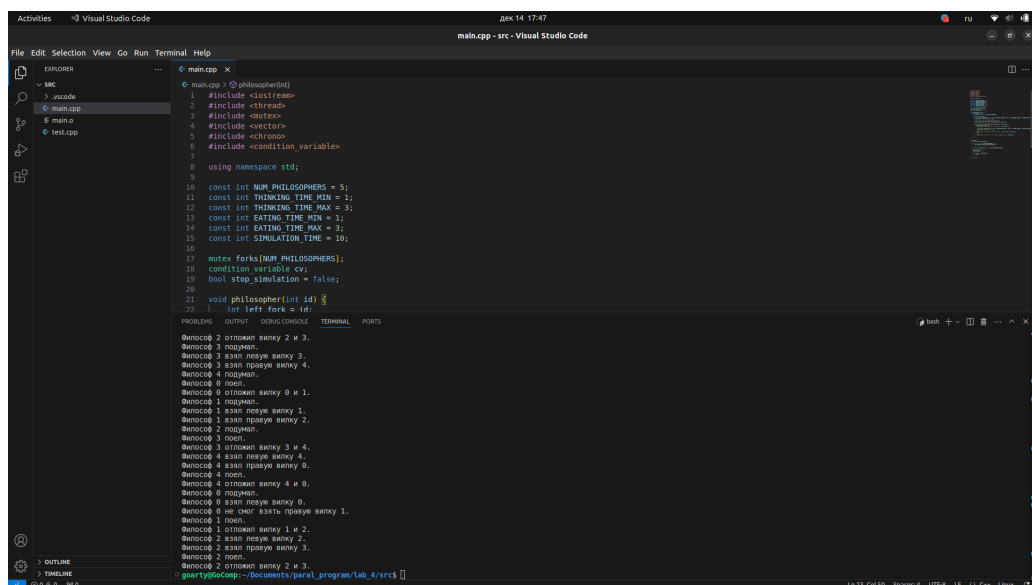
Философ 4 взял левую вилку 4.  
Философ 4 взял правую вилку 0.  
Философ 1 поел.  
Философ 1 отложил вилку 1 и 2.  
Философ 2 взял левую вилку 2.  
Философ 2 взял правую вилку 3.  
Философ 0 подумал.  
Философ 4 поел.  
Философ 4 отложил вилку 4 и 0.  
Философ 0 взял левую вилку 0.  
Философ 0 взял правую вилку 1.  
Философ 2 поел.  
Философ 2 отложил вилку 2 и 3.  
Философ 3 подумал.  
Философ 3 взял левую вилку 3.  
Философ 3 взял правую вилку 4.  
Философ 4 подумал.  
Философ 0 поел.  
Философ 0 отложил вилку 0 и 1.  
Философ 1 подумал.  
Философ 1 взял левую вилку 1.  
Философ 1 взял правую вилку 2.  
Философ 2 подумал.  
Философ 3 поел.  
Философ 3 отложил вилку 3 и 4.  
Философ 4 взял левую вилку 4.  
Философ 4 взял правую вилку 0.  
Философ 4 поел.  
Философ 4 отложил вилку 4 и 0.  
Философ 0 подумал.  
Философ 0 взял левую вилку 0.  
Философ 0 не смог взять правую вилку 1.  
Философ 1 поел.

Философ 1 отложил вилку 1 и 2.  
Философ 2 взял левую вилку 2.  
Философ 2 взял правую вилку 3.  
Философ 2 поел.  
Философ 2 отложил вилку 2 и 3.

## 3 Заключение

В данной работе я изучил возможности языка C++ в работе с библиотекой thread и mutex, а именно научился с помощью мьютексов накладывать блокировки в потоках.

## 4 Результат запуска



```
main.cpp - src - Visual Studio Code
dek 14 17:47

File Edit Selection View Go Run Terminal Help
EXPLORER
src
  .vscode
  main.cpp
  main0
  test.cpp
main.cpp
1 #include <iostream>
2 #include <thread>
3 #include <mutex>
4 #include <vector>
5 #include <chrono>
6 #include <condition_variable>
7
8 using namespace std;
9
10 const int NUM_PHILOSOPHERS = 5;
11 const int THINKING_TIME_MIN = 1;
12 const int THINKING_TIME_MAX = 3;
13 const int EATING_TIME_MIN = 1;
14 const int EATING_TIME_MAX = 3;
15 const int SIMULATION_TIME = 10;
16
17 mutex forks[NUM_PHILOSOPHERS];
18 condition_variable cv;
19 bool stop_simulation = false;
20
21 void philosopher(int id) {
22     int left_fork = id;
23     int right_fork = (id + 1) % NUM_PHILOSOPHERS;
24     while (true) {
25         cout << "Философ " << id << " берет левую вилку " << left_fork << endl;
26         cv.wait(forks[left_fork]);
27         cout << "Философ " << id << " берет правую вилку " << right_fork << endl;
28         cv.wait(forks[right_fork]);
29         cout << "Философ " << id << " ест " << endl;
30         auto start = chrono::high_resolution_clock::now();
31         auto end = start + chrono::seconds(rand() % 3 + 1);
32         cv.wait_for(end, cv);
33         cout << "Философ " << id << " кладет правую вилку " << right_fork << endl;
34         cv.notify_one();
35         cout << "Философ " << id << " кладет левую вилку " << left_fork << endl;
36         cv.notify_one();
37     }
38 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Философ 2 отложил вилку 2 и 3.
Философ 3 поел.
Философ 3 взял левую вилку 3.
Философ 3 взял правую вилку 4.
Философ 4 поел.
Философ 0 поел.
Философ 0 отложил вилку 0 и 1.
Философ 1 поел.
Философ 1 взял левую вилку 1.
Философ 1 взял правую вилку 2.
Философ 2 поел.
Философ 3 поел.
Философ 3 отложил вилку 3 и 4.
Философ 4 взял левую вилку 4.
Философ 4 взял правую вилку 0.
Философ 4 поел.
Философ 4 отложил вилку 4 и 0.
Философ 0 поел.
Философ 0 взял левую вилку 0.
Философ 0 не смог взять правую вилку 1.
Философ 1 поел.
Философ 1 отложил вилку 1 и 2.
Философ 2 взял левую вилку 2.
Философ 2 взял правую вилку 3.
Философ 2 поел.
Философ 2 отложил вилку 2 и 3.
```