

Задание 1

Необходимые знания

1. Компилирование программ с помощью gcc.
2. Состояние гонки.
3. Критическая секция.
4. POSIX threads: как создавать, как дожидаться завершения.
5. Как линковаться на библиотеку `pthread`

Скомпилировать `mutex.c` без использования и с использованием мьютекса. Объяснить разницу в поведении программы.

Сборка без мьютекса:

```
ubuntu@ubuntu:~/osis/lab5/src$ ./mutex
doing one thing
counter = 0
doing another thing
counter = 0
doing one thing
counter = 1
doing another thing
counter = 1
doing one thing
counter = 2
doing another thing
counter = 2
doing one thing
counter = 3
doing another thing
counter = 3
doing one thing
counter = 4
doing another thing
counter = 4
doing one thing
counter = 5
doing another thing
counter = 5
doing one thing
```

.....

```
doing another thing
counter = 46
doing one thing
counter = 47
doing another thing
counter = 47
doing one thing
counter = 48
doing another thing
counter = 48
doing one thing
counter = 49
doing another thing
counter = 50
} All done, counter = 51
```

Сборка уже с мьютексом:

```
ubuntu@ubuntu:~/osis/lab5/src$ ./mutex
doing one thing
counter = 0
doing one thing
counter = 1
doing one thing
counter = 2
doing one thing
counter = 3
.....
```

```
} counter = 90
  doing another thing
  counter = 91
  doing another thing
  counter = 92
  doing another thing
  counter = 93
  doing another thing
  counter = 94
  doing another thing
  counter = 95
  doing another thing
  counter = 96
  doing another thing
  counter = 97
  doing another thing
  counter = 98
  doing another thing
  counter = 99
} All done, counter = 100
ubuntu@ubuntu:~/osis/lab5/src$
```

Мьютекс – это примитив синхронизации, который решает проблему **гонки данных** путем блокировок потоков.

Если запустить программу несколько раз, то можно будет увидеть, что программа возвращает разный результат. С маленьким это не сильно видно, но если увеличить counter к примеру до 100000, то расхождения будут видны более явно.

Задание 2

Необходимые знания

1. POSIX threads: как создавать, как дожидаться завершения.
2. Как линковаться на библиотеку `pthread`
3. Как использовать мьютексы.

Написать программу для параллельного вычисления факториала по модулю `mod` (`k!`), которая будет принимать на вход следующие параметры (пример: `-k 10 --pnum=4 --mod=10`):

1. `k` - число, факториал которого необходимо вычислить.
2. `pnum` - количество потоков.
3. `mod` - модуль факториала

Для синхронизации результатов необходимо использовать мьютексы.

```
ubuntu@ubuntu:~/osis/lab5/src$ gcc -pthread factmod.c -o factmod
ubuntu@ubuntu:~/osis/lab5/src$ ./factmod -k 10 --pnum=4 --mod=100
Elapsed time: 1.165000ms
Factorial of 10 mod 100 is: 0
ubuntu@ubuntu:~/osis/lab5/src$
```

Задание 3

Необходимые знания

1. Состояние deadlock

Напишите программу для демонстрации состояния deadlock.

```
ubuntu@ubuntu:~/osis/lab5/src$ gcc -pthread deadlock.c -o deadlock
ubuntu@ubuntu:~/osis/lab5/src$ ./deadlock
Thread 2: захватил lock2
Thread 1: захватил lock1
Thread 1: пытается захватить lock2...
Thread 2: пытается захватить lock1...
^C
ubuntu@ubuntu:~/osis/lab5/src$
```

Тут я явно показал ситуацию с deadlock, программа не завершится, если только ее не остановить принудительно.