

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент Программной Инженерии

**МИКРОПРОЕКТ 2**

Пояснительная записка

Выполнил студент группы БПИ196 (2 подгруппа)

Шестаков Михаил Сергеевич

**Вариант 6 (28 в списке,  $28 \bmod 22 = 6$ ).**

*Задача о курильщиках.*

Есть три процесса-курильщика и один процесс-посредник. Курильщик непрерывно скручивает сигареты и курит их. Чтобы скрутить сигарету, нужны табак, бумага и спички. У одного процесса-курильщика есть табак, у второго – бумага, а у третьего – спички. Посредник кладет на стол по два разных случайных компонента. Тот процесс-курильщик, у которого есть третий компонент, забирает компоненты со стола, скручивает сигарету и курит. Посредник дожидается, пока курильщик закончит, затем процесс повторяется. Создать многопоточное приложение, моделирующее поведение курильщиков и посредника. При решении задачи использовать семафоры.

## 1. Описание работы программы

В задаче получается, что курильщик ждет одновременно сразу два ресурса. Проблема в том, что поток не может одновременно ждать два семафора [1] (без хитрых схем, вроде получения значения семафора и в случае неудачи возвращения его обратно в семафор). Чтобы решить проблему, сделаем каждому курильщику по семафору, который будет говорить, что появились *два других* ресурса. Тогда поставщик, выложив на стол два ресурса, может увеличить значение семафора с третьим ресурсом на единицу, тем самым разбудив соответствующего курильщика.

На самом деле данная идея была предложена в ответе на [stackoverflow](#) [2], но я невнимательно его прочитал и решив, что он не подходит, придумывал её сам.\

Поскольку в `std` нормальные семафоры появляются только в C++ 20 [3] (который пока что ещё не совсем полноценно поддерживается), мною было принято решение писать всё с использованием `pthread` и `semaphore`, чтобы всё решение было написано в единообразном стиле.

Также в программе возникала проблема с несинхронным выводом. Но мне удалось найти достаточно удачное это решение, с использованием вывода внутри деструктора класса [4]. Оригинальное решение использует `mutex`, поэтому пришлось заменить их на семафоры, чтобы моё решение лучше соответствовало условию задачи. В синхронном выводе есть ещё один дополнительный плюс: мы можем избавиться от времени в выводе, поскольку все сообщения и так выводятся в правильном порядке.

Таким образом в программе используются следующие семафоры:

`sem_t resourcesWait[3]` -- семафоры, на которых курильщики ждут поступления ресурсов от поставщика

`sem_t smokeWait` -- семафор, на котором поставщик ждёт, пока курильщик докурит

`sem_t coutWait` -- семафор для синхронизации вывода

## 2. Аргументы командной строки

Мне показалось, что подход с консольными аргументами более удобен, чем последовательные запросы у пользователя. В программе используется следующий формат консольных аргументов:

```
./program <smokeTime> <iterationsCount>,
```

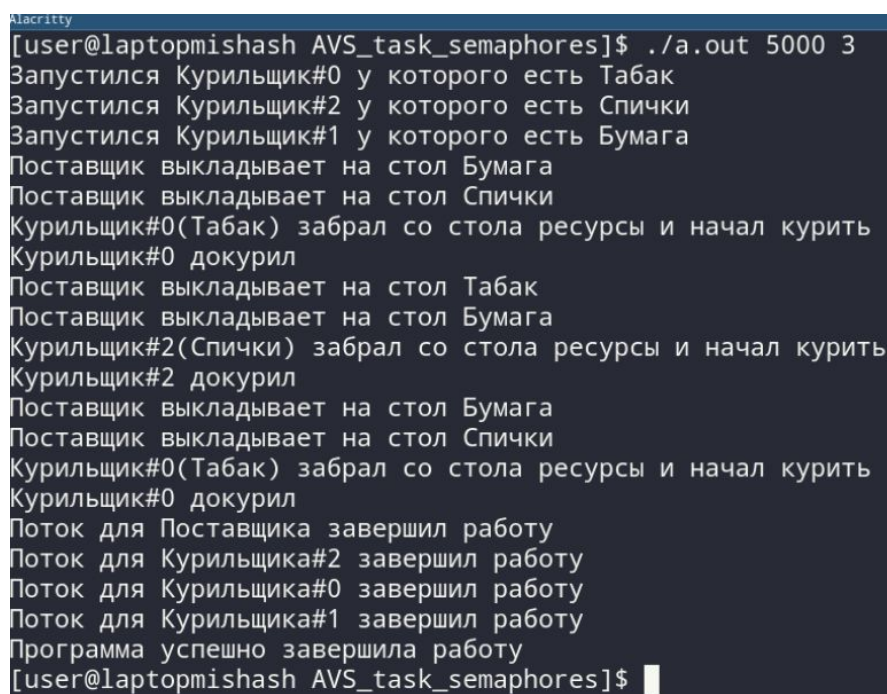
где `smokeTime` -- время курения курильщика,

`iterationsCount` -- количество итераций (сколько раз поставщик будет выкладывать компоненты для курения)

## 3. Тестирование

### 3.1. Простой тест

Запускаем программу с временем на курение 5000 и количеством итераций 3:



```
Macratty
[user@laptopmishash AVS_task_semaphores]$ ./a.out 5000 3
Запустился Курильщик#0 у которого есть Табак
Запустился Курильщик#2 у которого есть Спички
Запустился Курильщик#1 у которого есть Бумага
Поставщик выкладывает на стол Бумага
Поставщик выкладывает на стол Спички
Курильщик#0(Табак) забрал со стола ресурсы и начал курить
Курильщик#0 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Бумага
Курильщик#2(Спички) забрал со стола ресурсы и начал курить
Курильщик#2 докурил
Поставщик выкладывает на стол Бумага
Поставщик выкладывает на стол Спички
Курильщик#0(Табак) забрал со стола ресурсы и начал курить
Курильщик#0 докурил
Поток для Поставщика завершил работу
Поток для Курильщика#2 завершил работу
Поток для Курильщика#0 завершил работу
Поток для Курильщика#1 завершил работу
Программа успешно завершила работу
[user@laptopmishash AVS_task_semaphores]$
```

Можно посмотреть, как отрабатывают временные задержки в файле **test1\_out.gif**. (к сожалению, мне не удалось вставить gif-файл внутрь pdf)

### 3.2. Запуск программы без аргументов

```
[user@laptopmishash AVS_task_semaphores]$ ./a.out
Вадано некорректное количество аргументов
./program <smokeTime> <iterationsCount>
smokeTime -- время на курение
iterationsCount -- суммарное количество выкладываний на стол компонентов
[user@laptopmishash AVS_task_semaphores]$
```

Ожидаемо выводится сообщение о некорректных консольных аргументах и правильный формат.

### 3.3. Большой тест

Выставим задержку равной 0, и количество итераций 100000

```
[user@laptopmishash AVS_task_semaphores]$ ./a.out 0 100000 > test2_out.txt
[user@laptopmishash AVS_task_semaphores]$
```

Получился довольно большой файл (26 мегабайт). Посмотрим его с помощью vim, потому что он умеет хорошо справляться с файлами любого размера

```
[user@laptopmishash AVS_task_semaphores]$ vim test2_out.txt
```

```
Курильщик#0 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Спички
Курильщик#1(Бумага) забрал со стола ресурсы и начал курить
Курильщик#1 докурил
Поставщик выкладывает на стол Бумага
Поставщик выкладывает на стол Спички
Курильщик#0(Табак) забрал со стола ресурсы и начал курить
Курильщик#0 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Спички
Курильщик#1(Бумага) забрал со стола ресурсы и начал курить
Курильщик#1 докурил
Поставщик выкладывает на стол Бумага
Поставщик выкладывает на стол Спички
Курильщик#0(Табак) забрал со стола ресурсы и начал курить
Курильщик#0 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Спички
Курильщик#1(Бумага) забрал со стола ресурсы и начал курить
Курильщик#1 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Спички
Курильщик#1(Бумага) забрал со стола ресурсы и начал курить
Курильщик#1 докурил
Поставщик выкладывает на стол Бумага
"test2_out.txt" 400008L, 27800598B
```

7695,5-3

1%

Как видим, на данных строках всё корректно. Количество строк в файле тоже корректное: 400008. У нас выводится по 4 сообщения на каждый из 100000 циклов + 8 сообщений в начале и в конце.

3.6. Ещё один тест с небольшой временной задержкой и бОльшим, чем в самом первом количеством итераций:

```
[user@laptopmishash AVS_task_semaphores]$ ./a.out 100 30
```

```
Курильщик#1 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Спички
Курильщик#1(Бумага) забрал со стола ресурсы и начал курить
Курильщик#1 докурил
Поставщик выкладывает на стол Бумага
Поставщик выкладывает на стол Спички
Курильщик#0(Табак) забрал со стола ресурсы и начал курить
Курильщик#0 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Бумага
Курильщик#2(Спички) забрал со стола ресурсы и начал курить
Курильщик#2 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Бумага
Курильщик#2(Спички) забрал со стола ресурсы и начал курить
Курильщик#2 докурил
Поставщик выкладывает на стол Табак
Поставщик выкладывает на стол Спички
Курильщик#1(Бумага) забрал со стола ресурсы и начал курить
Курильщик#1 докурил
Поток для Поставщика завершил работу
Поток для Курильщика#0 завершил работу
Поток для Курильщика#1 завершил работу
Поток для Курильщика#2 завершил работу
Программа успешно завершила работу
[user@laptopmishash AVS_task_semaphores]$
```

(анимированную версию можно посмотреть в файле **test6\_out.gif**)

3.7. Некорректные данные – вместо числа в консольный аргумент передана некорректная строка

```
[user@laptopmishash AVS_task_semaphores]$ ./a.out test123 10
Некорректно задан числовой аргумент
./program <smokeTime> <iterationsCount>
smokeTime -- время на курение
iterationsCount -- суммарное количество выкладываний на стол компонентов
[user@laptopmishash AVS_task_semaphores]$
```



### 3.8. Некорректные данные – в время на курение передано недопустимое значение

```
[user@laptopmishash AVS_task_semaphores]$ ./a.out -1 10
Задано некорректное время на курение
./program <smokeTime> <iterationsCount>
smokeTime -- время на курение
iterationsCount -- суммарное количество выкладываний на стол компонентов
[user@laptopmishash AVS_task_semaphores]$ █
```

### 3.9. Некорректные данные – в количество итераций передано недопустимое значение

```
[user@laptopmishash AVS_task_semaphores]$ ./a.out 1000 -34124
Задано некорректное количество итераций
./program <smokeTime> <iterationsCount>
smokeTime -- время на курение
iterationsCount -- суммарное количество выкладываний на стол компонентов
[user@laptopmishash AVS_task_semaphores]$ █
```

## Источники

- [1] <https://stackoverflow.com/questions/826769/is-there-a-way-to-wait-on-multiple-semaphors>
- [2] <https://stackoverflow.com/questions/7780749/waiting-on-multiple-semaphores-without-busy-waiting-c-c-linux>
- [3] [https://en.cppreference.com/w/cpp/thread/counting\\_semaphore](https://en.cppreference.com/w/cpp/thread/counting_semaphore)
- [4] <https://stackoverflow.com/questions/2196155/is-there-anyway-to-write-the-following-as-a-c-macro>