Национальный исследовательский университет
«Высшая школа экономики»
Факультет компьютерных наук
Образовательная программа «Программная инженерия»

Микро-проект по дисциплине
«Архитектура вычислительных систем»

Выполнил: студент группы БПИ199
Шарипов Сардор Уткирович

Вариант 5

Задача о каннибалах. Племя из n дикарей ест вместе из большого горшка, который вмещает m кусков тушеного миссионера. Когда дикарь хочет обедать, он ест из горшка один кусок, если только горшок не пуст, иначе дикарь будит повара и ждет, пока тот не наполнит горшок. Повар, сварив обед, засыпает. Создать многопоточное приложение, моделирующее обед дикарей. При решении задачи использовать семафоры.

Описание принципа работы программы

В данной задаче используется многопоточность, то есть в нашей задачи потоки — это дикари, когда в горшке есть мясо первый потянувший в горшок дикарь берет мясо и кушает (т. е. первый успевший поток берет мясо) для остальных дикарей мясо недоступно во время того, когда дикарь берет мясо (т. е один поток берет остальные ждут). Чтобы реализовать это потребуется использование мьютексов, способ блокировки критической секции. В критической секции только 1 поток имеет “власть”, а остальные ждут. А что, если мясо закончилось. Тогда тот дикарь чья очередь была забрать мясо с горшка будит повара, то есть в нашей программе этот поток, который в критической секции будит другой поток, который наполняет горшок мясом.

Критическая секция в программе — это метод eating.

```
void eating(int indexOfThread) {
    mutex.lock(); // Открываем горшок лишь одному дикарю
    currentIndexOfSavage %= numberOfSavage; // индекс этого дикаря
    countOfMeatLeft -= 1; // Дикарь скушал один кусок

    currentIndexOfSavage += 1;
    std::cout << "Поток № " << indexOfThread << ". "
              << "Дикарь №" << currentIndexOfSavage << " "
              << "скушал один кусок.";
    if (countOfMeatLeft == 0)
        std::cout << "Не осталось мяса в горшке" << std::endl;
    else
        std::cout << "Осталось " << countOfMeatLeft << " "
                  << "мяса в горшке" << std::endl;

    // при опустошение горшка повар заполняет его
    if (countOfMeatLeft == 0) {
        auto *chiefThread = new std::thread(fillPot);
        chiefThread->join();
        delete chiefThread;
    }
    mutex.unlock(); // Закрываем наш горшок и чтобы другой дикарь смог подойти
}
```

А метод, когда повар заполняет горшок это

```
void fillPot() {
    countOfMeatLeft = maxCountOfMeat;
    std::cout << "Повар проснулся и наполнил горшок "
              << countOfMeatLeft << " кусками мяса."
              << std::endl << std::endl;
}
```

Описание входных данных

Входные данные должны быть в аргументах программы формата

<кол-во дикарей> <кол-во кусков мяса> <кол-во повторений программы>

Последний аргумент нужен для того, чтобы не ввести программу в бесконечный цикл, а повторить какое-то кол-во раз (то есть накормить дикарей какое-то кол-во раз).

Все входные аргументы должны быть целочисленные, в пределах

[0; Int32.Max]

При неправильном входным данным программа выдаст сообщение об ошибке и закончит программу.

Описание выходных данных

В качестве выходных данных пользователь получает полный процесс работы программы, с результатами работы каждого потока.

```
sardor@sharipovPC:/mnt/f/Прога/Assembler/Projects/MP2$ ./a.out 3 2 2
Поток № 1. Дикарь №1 скушал один кусок.Осталось 1 мяса в горшке
Поток № 2. Дикарь №2 скушал один кусок.Не осталось мяса в горшке
Повар проснулся и наполнил горшок 2 кусками мяса.

Поток № 3. Дикарь №3 скушал один кусок.Осталось 1 мяса в горшке
Поток № 1. Дикарь №1 скушал один кусок.Не осталось мяса в горшке
Повар проснулся и наполнил горшок 2 кусками мяса.

Поток № 2. Дикарь №2 скушал один кусок.Осталось 1 мяса в горшке
Поток № 3. Дикарь №3 скушал один кусок.Не осталось мяса в горшке
Повар проснулся и наполнил горшок 2 кусками мяса.

Работа программы: 3.4153 ms
```

Также проверим на неправильные входные данные

```
sardor@sharipovPC:/mnt/f/Прога/Assembler/Projects/MP2$ ./a.out 0 1 1
Все значение должно быть > 0
```

```
sardor@sharipovPC:/mnt/f/Прога/Assembler/Projects/MP2$ ./a.out фффф 2 2
НЕ ЧИСЛО БЫЛО ПОДАНО
```

```
sardor@sharipovPC:/mnt/f/Прога/Assembler/Projects/MP2$ ./a.out 2 2 11111111111111111111111111111111111111111111
NOT IN RANGE  OF INT
```

```
sardor@sharipovPC:/mnt/f/Прога/Assembler/Projects/MP2$ ./a.out 2 2
Неправильное кол-во аргументов
Формат: <Кол-во Дикарей> <Кол-во Кусков Мяса> <Кол-во повторений>
```

Источники

https://ru.wikipedia.org/wiki/Мьютекс