ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии

ЗАДАЧА О КАННИБАЛАХ (вариант 5)

Отчёт

Дисциплина: «Архитектура вычислительных систем»

Исполнитель:
студент группы БПИ198
Баранов Г. А.

Москва 2020

СОДЕРЖАНИЕ

1. ТЕКСТ ЗАДАНИЯ

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 пользоваться семафорами.

2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Исполнение программы начинается с ввода двух параметров через командную консоль. Первый параметр – количество еды, второй – количество каннибалов. После этого создаются потоки в указанных количествах. В каждом потоке запускается 10 секундный таймер, который ограничивает время работы программы.

В методе для каннибалов используется семафор, который ограничивает количество каннибалов, которые за раз могут взять себе кусок еды. Сами каннибалы смотрят, есть ли еда и, если она есть, то уменьшают переменную с ее количеством на один. После чего они освобождают семафор и уходят в сон на несколько секунд. Также используется мьютекс для корректного вывода данных об оставшейся еде.

Повар работает постоянно, его задача проверять, осталась ли еда. Если еды больше нет, он восполняет переменную. После каждой проверки шеф уходит в сон на 1 секунду.

3. ТЕСТИРОВАНИЕ ПРОГРАММЫ

```
C:\Users\HYPERPC\source\repos\ConsoleApplication2\ConsoleApplication2>Cannibals 10 7
Cannibal 0 is eating at [0] Food left: 9
Cannibal 2 is eating at [0] Food left: 8
Cannibal 1 is eating at [0] Food left: 7
Cannibal 3 is eating at [0] Food left: 6
Cannibal 4 is eating at [0] Food left: 5
Cannibal 5 is eating at [0] Food left: 4
Cannibal 6 is eating at [0] Food left: 3
Cannibal 5 is eating at [1] Food left: 2
Cannibal 2 is eating at [1] Food left: 1
Cannibal 0 is eating at [1] Food left: 0
Food cooked!
Cannibal 5 is eating at [2] Food left: 9
Cannibal 2 is eating at [2] Food left: 8
Cannibal 3 is eating at [2] Food left: 7
Cannibal 4 is eating at [2] Food left: 6
Cannibal 0 is eating at [2] Food left: 5
Cannibal 1 is eating at [3] Food left: 4
Cannibal 6 is eating at [3] Food left: 3
Cannibal 2 is eating at [4] Food left: 2
Cannibal 5 is eating at [4] Food left: 1
Cannibal 0 is eating at [4] Food left: 0
Food cooked!
```

Рисунок 1 – Программа работает корректно, потоки чередуются, вывод оформлен правильно

```
Cannibal 6 is eating at [8] Food left: 9
Cannibal 4 is eating at [8] Food left: 8
Cannibal 0 is eating at [8] Food left: 7
Cannibal 5 is eating at [8] Food left: 6
Cannibal 2 is eating at [8] Food left: 5
Cannibal 3 is eating at [9] Food left: 4
Cannibal 1 is eating at [9] Food left: 3
Cannibal 4 is eating at [9] Food left: 2
Cannibal 6 is eating at [9] Food left: 1
Cannibal 1 is eating at [10] Food left: 0

C:\Users\HYPERPC\source\repos\ConsoleApplication2\ConsoleApplication2>
```

Рисунок 2 – Программа завершает работу корректно, в данном случае каннибалы успели доесть мясо, однако по времени повар уже не успел проснуться

```
C:\Users\HYPERPC\source\repos\ConsoleApplication2\ConsoleApplication2>Cannibals 0 0
Cannibal 0 is eating at [0] Food left: 0
Food cooked!
Cannibal 0 is eating at [1] Food left: 0
Food cooked!
Cannibal 0 is eating at [2] Food left: 0
Food cooked!
Cannibal 0 is eating at [4] Food left: 0
Food cooked!
Cannibal 0 is eating at [5] Food left: 0
Food cooked!
Cannibal 0 is eating at [6] Food left: 0
Food cooked!
Cannibal 0 is eating at [8] Food left: 0
Food cooked!
Cannibal 0 is eating at [10] Food left: 0
Food cooked!
```

Рисунок 3 – При некорректных данных программа установит значения 1, 1 по умолчанию

ИСТОЧНИКИ

1. SoftCraft, сайт по учебной дисциплине. [Электронный ресурс] http://softcraft.ru/ (дата обращения: 10.12.2020).
2. ProgrammerSought сайт по установке pthread. [Электронный ресурс] https://www.programmersought.com/article/71931627060/ (дата обращения: 10.12.2020)

ПРИЛОЖЕНИЕ 1

КОД ПРОГРАММЫ

```
#include <iostream>
//Без этого pthread не заработал
#define HAVE_STRUCT_TIMESPEC
#include <pthread.h>
#include <semaphore.h>
#include <windows.h>
#include <chrono>
#include <string>
#include <vector>

//Без этого pthread не заработал
#pragma comment(lib,"pthreadVC2.lib")

using namespace std;

int foodLeft;
int cannibalNumber;
int foodNumber;
sem_t semaphore;
pthread_mutex_t allowEat;

/*
Метод для того, чтобы каннибалы ели
*/
void* Cannibal(void* eater)
{
	//Начало и конец по времени
	auto start = chrono::system_clock::now();
	auto end = std::chrono::system_clock::now();
	//Номер каннибала
	int num = *((int*)eater);
	//Трапеза продолжается 10 секунд
	while ((std::chrono::duration_cast<std::chrono::seconds>(end - start).count() <=
10)) {
		//Уменьшаем значение семафора
		sem_wait(&semaphore);
		if (foodLeft > 0)
		{
			//Блокируем вывод, чтобы текст не бегал
			pthread_mutex_lock(&allowEat);
			cout << "Cannibal " << num << " is eating!";
			foodLeft--;
			cout << "Food left: " << foodLeft << endl;
			//Разблокируем обратно
			pthread_mutex_unlock(&allowEat);
		}
		//Семафор прибавляется
		sem_post(&semaphore);
		srand(time(0));
		//Каннибал засыпает на рандомное время
		int delay = 1000 + (rand() % 10) * 100;
		Sleep(delay);
		end = std::chrono::system_clock::now();
	}
	return NULL;
}

//Метлд в котором шеф готовит еду
void* Chief(void* args)
{
```

```
	//Шеф работает столько же сколько и едят каннибалы
	auto start = chrono::system_clock::now();
	auto end = std::chrono::system_clock::now();
	int meatAmount = *((int*)args);
	while ((std::chrono::duration_cast<std::chrono::seconds>(end - start).count() <=
10)) {
		//Проходим по семафору и ждем пока еды не останется
		if (foodLeft == 0)
		{
			//Заполняем еду
			foodLeft = meatAmount;
			cout << "Food cooked!" << endl;
		}
		Sleep(1000);
		end = std::chrono::system_clock::now();
	}
	return NULL;
}

int main(int argc, char* argv[])
{
	//Проверка входных аргументов
	if (argc != 3)
	{
		cout << "Incorrect amount of parametrs, please enter amount of cannibals
and food amount" << endl;
		return -1;
	}
	try {
		foodNumber = stoi(argv[1]);
		if (foodNumber <= 0) { foodNumber = 1; }
		foodLeft = foodNumber;
		cannibalNumber = stoi(argv[2]);
		if (cannibalNumber <= 0) { cannibalNumber = 1; }
	}
	catch (exception e) {
		cout << "Incorrect data";
		return -1;
	}
	//Инициализация мутексов и семафоров
	sem_init(&semaphore, 0, foodNumber);
	pthread_mutex_init(&allowEat, nullptr);
	int* arr = new int[cannibalNumber];
	vector<pthread_t> cannibals(cannibalNumber);
	pthread_t chief;
	//Создаем поток шефа
	pthread_create(&chief, NULL, Chief, &foodNumber);
	//Создаем каннибалов
	for (int t = 0; t < cannibalNumber; t++)
	{
		arr[t] = t;
		int rc = pthread_create(&cannibals[t], NULL, Cannibal, &arr[t]);
		if (rc)
		{
			printf("ERROR:return code from pthread_create() is %d\n", rc);
			exit(-1);
		}
	}
	//После завершения работы соединяем все потоки
	for (int t = 0; t < cannibalNumber; t++)
	{
		pthread_join(cannibals[t], NULL);
	}
	delete[] arr;
```

```
        pthread_join(chief, NULL);
        sem_destroy(&semaphore);
        pthread_exit(NULL);
}
```