ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Отчет по домашнему заданию по дисциплине "Архитектура вычислительных систем"

Пояснительная Записка

Исполнитель:
Студент группы БПИ195(1)
Ни Эдуард
17.11.2020 г.

Москва 2020

Условие задания

ВАРИАНТ23. *Первая военная задача.* Темной-темной ночью прапорщики Иванов, Петров и Нечепорчук занимаются хищением военного имущества со склада родной военной части. Будучи умными людьми и отличниками боевой и строевой подготовки, прапорщики ввели разделение труда: Иванов выносит имущество со склада, Петров грузит его в грузовик, а Нечепорчук подсчитывает рыночную стоимость добычи. Требуется составить многопоточное приложение, моделирующее деятельность прапорщиков. При решении использовать парадигму «производитель-потребитель».

Решение

Для решения задачи будем пользоваться парадигмой конструирования многопоточных приложений **производители и потребители**.

Производители и потребители *– это парадигма взаимодействующих неравноправных потоков. Одни потоки «производят» данные, другие их «потребляют». Часто такие потоки организуются в конвейер, через который проходит информация. Каждый поток конвейера потребляет выход своего предшественника и производит входные данные для своего последователя. Другой распространенный способ организации потоков – древовидная структура или сети слияния, на этом основан, в частности, метод дихотомии.*

По условию, мы имеем трех прапорщиков, которые расхищают склад с военным имуществом. Прапорщик Иванов по нашей парадигме выносит военное имущество (подготавливает для Петрова). Прапорщик Петров загружает машину вынесенным оборудованием (подготавливает для Нечепорчука). Нечепорчук, сидя в машине, подсчитывает рыночную стоимость загруженного Петровым оборудования.

Составление программы

Для работы программы будем использовать библиотеки для языка C++: iostream, unistd.h, pthread.h, semaphore.h.

Наша программа будет работать бесконечно.

Москва 2020

Текст программы

```
#include <iostream>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>

const int munSize = 10;    //Количество военного имущества.
int cntr = 0;   //Количество вынесеного имущества.
int cost = 0;    //Суммманая стоимость вынесенного имущества.
int munition[munSize];  //Военное имущество.
int delCost = 4;    //Делитель Стоимости одного оборудования на рынке.
int front = 0; //Индекс для чтения из буффера.
int rear = 0;   //Индекс для записи в буффер.
sem_t empty; //семафор - буффер пуст
sem_t full; //семафор - буффер полон
pthread_mutex_t mutexW; //mutex для записи
pthread_mutex_t mutexR; //mutex для чтения
pthread_mutex_t mutexC; //mutex для счетчика

/**
 * Метод передается в поток в качестве производителя.
 * @param mun Идентификатор производителя.
 */
void* Producer(void* mun);

/**
 * Метод передается в поток в качестве потребителя.
 * @param mun Идентификатор потребителя.
 */
void* Consumer(void* mun);

/**
 * Метод передается в поток в качестве счетчика срабатываний производителя-потребителя.
 * Работает аналогично потребителю.
 * @param mun Идентификатор счетчика.
 */
void* Counter(void* mun);

int main() {
    //Инициализируем мутексы.
    pthread_mutex_init(&mutexW, nullptr);
    pthread_mutex_init(&mutexR, nullptr);
    pthread_mutex_init(&mutexC, nullptr);
    //Семафоры.
    sem_init(&empty, 0, munSize);
    sem_init(&full, 0, 0);

    //Пускаем производителя - выносит оборудование из склада.
    pthread_t threadProd;
    std::string pName = "Иванов";
    pthread_create(&threadProd, nullptr, Producer, (void*)(&pName));

    //Пускаем потребителя - загружает в машину.
    pthread_t threadCons;
    std::string cName = "Петров";
    pthread_create(&threadCons, nullptr, Consumer, (void*)(&cName));

    //Главный поток - подсчет количества вынесенного оборудования.
    std::string cntrName = "Нечепорчук";
    Counter((void*)&(cntrName));
```

Москва 2020

```
    return 0;
}

void* Producer(void* mun) {
    const char* pName = (*((std::string*) mun)).c_str();
    while (true) {
        int data = rand() % 10000; //Генерируем номер оборудования
        pthread_mutex_lock(&mutexW);    //Закрываем мутекс.
        sem_wait(&empty);   //Количество свободных ячеек увеличилось на 1
        munition[rear] = data;
        rear = (rear + 1) % munSize;
        sem_post(&full);    //количество занятых ячеек уменьшить на 1
        pthread_mutex_unlock(&mutexW);  //Открываем мутекс.
        printf("Прапорщик %s выносит имущество № [%d] со склада.\n", pName, data);

        sleep(2);
    }
}

void* Consumer(void* mun) {
    const char* cName = (*((std::string*) mun)).c_str();
    int res;
    while (true) {
        pthread_mutex_lock(&mutexR);    //Закрываем мутекс.
        sem_wait(&full);    //количество занятых ячеек уменьшить на 1
        res = munition[front];
        front = (front + 1) % munSize;
        sem_post(&empty);   //Количество свободных ячеек увеличилось на 1
        pthread_mutex_unlock(&mutexR);  //Открываем мутекс.
        printf("Прапорщик %s грузит оборудование № [%d] в грузовик.\n", cName, res);
        ++cntr;
        sleep(4);
    }
}

void* Counter(void* mun) {
    const char* cntrName = (*((std::string*) mun)).c_str();

    while (true) {
        pthread_mutex_lock(&mutexC);
        sem_wait(&full);

        if (cntr > munSize & (cntr - 1) % munSize == 0)
            printf("Оборудования там больше, чем казалось... Берем все!\n");
        sem_post(&empty);
        pthread_mutex_unlock(&mutexC);
        cost += munition[cntr % munSize - 1] / delCost;  //Суммируем стоимость вынесенного
оборудования.

        printf("Прапорщик %s подчитал, что было вынесено оборудования (%dшт.) на сумму %d
у.е.\n", cntrName, cntr,
            cost);
        sleep(4);
    }
}
```

Описание программы

1. Глобальные переменные.

```
const int munSize = 10;    //Количество военного имущества.
int cntr = 0;   //Количество вынесеного имммущества.
int cost = 0;    //Суммманая стоимость вынесенного имущества.
int munition[munSize];  //Военное имущество.
int delCost = 4;    //Делитель Стоимости одного оборудования на рынке.
int front = 0; //Индекс для чтения из буффера.
int rear = 0;   //Индекс для записи в буффер.
sem_t empty; //семафор - буффер пуст
sem_t full; //семафор - буффер полон
pthread_mutex_t mutexW; //mutex для записи
pthread_mutex_t mutexR; //mutex для чтения
pthread_mutex_t mutexC; //mutex для счетчика
```

Рисунок 1 – Глобальные переменные

Задаем значение для количества вынесенного имущества *munSize* и массив, в которое будем вносить *int munition[munSize]*. Инициализируем два индекса, указывающие на чтение буфера и записи – *front* и *rear*. Создаем 3 мутекса для записи, чтения и счетчика и два семафора. Мутексы и семафоры используются для предотвращения ситуаций, когда поток1 еще не завершил работу, а поток2 работает с данными из потока1.

2. Метод производителя Producer.

```
void *Producer(void *mun) {
    const char *pName = (*((std::string *) mun)).c_str();
    while (true) {
        int data = rand() % 10000; //Генерируем номер оборудования
        pthread_mutex_lock(&mutexW);    //Закрываем мутекс.
        sem_wait(&empty);   //Количество свободных ячеек увеличилось на 1
        munition[rear] = data;
        rear = (rear + 1) % munSize;
        sem_post(&full);    //количество занятых ячеек уменьшить на 1
        pthread_mutex_unlock(&mutexW);  //Открываем мутекс.
        printf("Прапорщик %s выносит имущество № [%d] со склада.\n", pName, data);

        sleep(2);
    }
}
```

Рисунок 2 – Метод производителя

Москва 2020

Метод представляет собой производителя(ей), который подготавливает данные для потребителя(ей). В аргументах метода переменная void *mun нужно для идентификации потребителя. Через разыменование получаем строку – имя. Далее в бесконечном цикле генерируем с помощью rand() номер оборудования, в диапазоне от 1 до 10000. Закрываем мутекс и показываем, что количество свободных ячеек увеличилось на 1. Записываем в украденное оборудование новое: munition[rear] = data. Открываем мутекс. Выводим сообщение о том, что прапорщик выносит имущество под номером data со склада, приостанавливаем поток на 2 секунды.

3. Метод потребителя Consumer.

```
void *Consumer(void *mun) {
    const char *cName = (*((std::string *) mun)).c_str();
    int res;
    while (true) {
        pthread_mutex_lock(&mutexR);    //Закрываем мутекс.
        sem_wait(&full);    //количество занятых ячеек уменьшить на 1
        res = munition[front];
        front = (front + 1) % munSize;
        sem_post(&empty);   //Количество свободных ячеек увеличилось на 1
        pthread_mutex_unlock(&mutexR);  //Открываем мутекс.
        printf("Прапорщик %s грузит оборудование № [%d] в грузовик.\n", cName, res);
        ++cntr;
        sleep(4);
    }
}
```

Рисунок 3 – Метод Consumer

Метод представляет собой потребителя(ей), который работает с данными, подготовленными производителем. В аргументах метода переменная void *mun нужно для идентификации потребителя. Через разыменование получаем строку – имя. Далее в бесконечном цикле получаем добавленное производителем оборудование и выводим сообщение о том, что прапорщик грузит оборудование res в грузовик, приостанавливаем поток на 4 секунды.

4. Метод счетчика Counter.

```
void *Counter(void *mun) {
    const char *cntrName = (*((std::string *) mun)).c_str();

    while (true) {
        pthread_mutex_lock(&mutexC);
        sem_wait(&full);

        if (cntr > munSize & (cntr - 1) % munSize == 0)
            printf("Оборудования там больше, чем казалось... Берем все!\n");
        sem_post(&empty);
        pthread_mutex_unlock(&mutexC);
        cost += munition[cntr % munSize - 1] / delCost;  //Суммируем стоимость вынесенного оборудования.

        printf("Прапорщик %s подчитал, что было вынесено оборудования (%dшт.) на сумму %d у.е.\n", cntrName, cntr,
               cost);
        sleep(4);
    }
}
```

Рисунок 5 – Метод Counter

Работает почти также, как и Consumer. Представляет собой потребителя(ей), который работает с данными, который подготовил прошлый потребитель (производитель для Consumer). В аргументах метода переменная void *mun нужно для идентификации потребителя. Через разыменование получаем строку – имя. Далее в бесконечном цикле прибавляем к переменной cost стоимость оборудования, которое занес в машину прошлый прапорщик (munition[cntr % munSize - 1] = = munition[front - 1]). Стоимость военного имущества расчитываем как номер имущества делить на delCost, delCost = 4. Выводим сообщение о том, что прапорщик посчитал стоимость всего оборудования, которое было загружено в машину, останавливаем поток на 4 секунды.

Тестирование программы

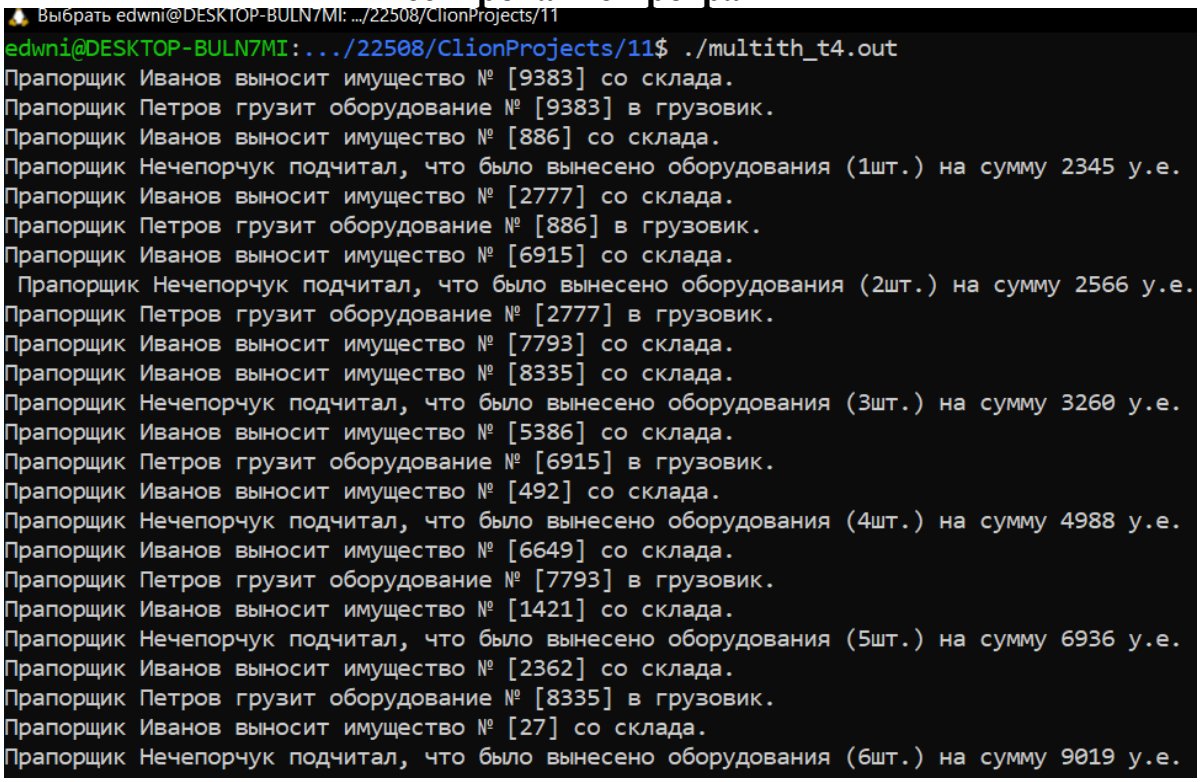

Рисунок 6- результат выполнения 1

Как видно из (рис. 6), программа работает корректно. Прапорщик Иванов выносит имущество под номером N со склада, а Иванов кладет его в грузовик. Нечепорчук подсчитывает количество вынесенных объектов и сумму в условных единицах. Нужно заметить, что Нечепорчук фиксирует результаты только действия прапорщика Иванова.

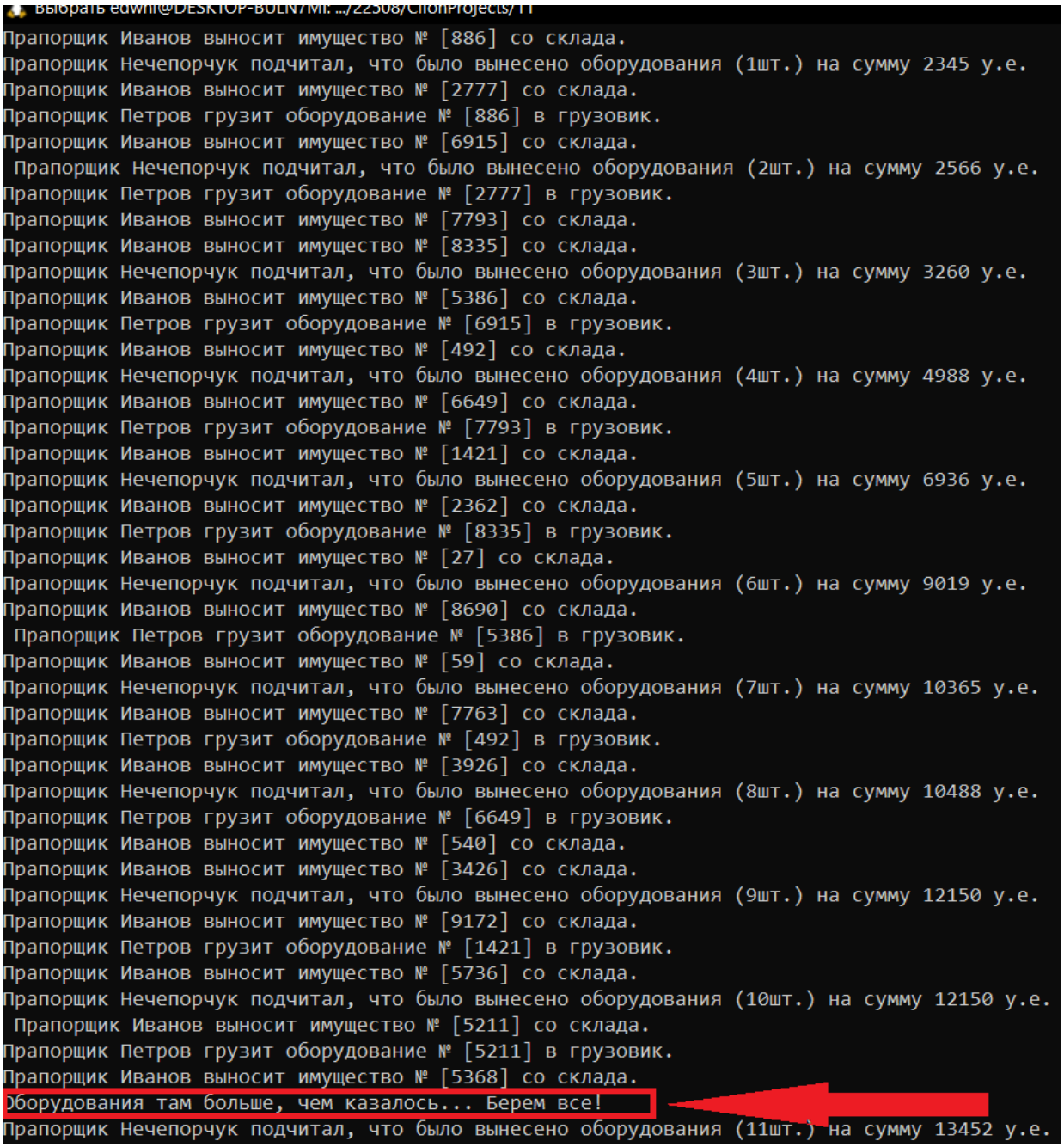

Рисунок 7 - результат выполнения 2

По (рис. 7) видно: так как мы ограничили количество грузов до 10, программа должна выполняться бесконечно. Поэтому каждый кратный 10ти раз мы будем выводить сообщение о том, что оборудования больше, чем, было заявлено.

Используемые источники

Москва 2020

1. SoftCraft, Архитектура вычислительных систем. Многопоточность [Электронный ресурс] //URL: http://www.softcraft.ru/edu/comparch/lect/07-parthread/multitreading.pdf, свободный. (дата обращения 12.11.2020, режим доступа: свободный).
2. SoftCraft, Архитектура вычислительных систем. Многопоточное программирование.Синхронизация [Электронный ресурс] //URL: http://www.softcraft.ru/edu/comparch/practice/thread/02-sync/, свободный. (дата обращения 12.11.2020, режим доступа: свободный).
3. ИНТУИТ. Операционные системы – аспекты параллелизма. [Электронный ресурс] //URL: https://intuit.ru/studies/courses/4447/983/lecture/14923?page=4, свободный. (дата обращения 12.11.2020, режим доступа: свободный).

Москва 2020