**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА (САМАРСКИЙ УНИВЕРСИТЕТ)»**

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ

Кафедра программных систем

Дисциплина

**Технологии промышленного программирования**

# ОТЧЕТ

по лабораторной работе

## Запуск и синхронизация нитей

Вариант № 2.1

Студент: Лазарев М.Ю

Группа: 6232-020402D

Преподаватель: Баландин А.В.

Самара 2025

# 1 Задание

Цель работы - освоение функций запуска и синхронизации нитей при разработке многонитиевых приложений в ОСРВ QNX.

Разработать приложение, состоящее из одного процесса с тремя запущенными нитями:

1. M(main).
2. T1(F1).
3. T2(F2).

В качестве нити M(main) выступает функция main(). Нити T1(F1) и T2(F2) запускаются нитью M(main) на базе соответственно функций F1() и F2(). Все три нити, работая параллельно, должны совместно динамически сформировать текст вида:

"Text0, Text1, Text2.\n"

Вначале нить M(main) запускает нить T1(F1), передавая ей в качестве параметра указатель совместно формируемого текста и функцию F2.

Далее нить M(main), записывая в текст букву за буквой, формирует свою часть текста:

"Text0, "

После формирования нитями всего текста нить M(main) выдаёт его на печать и завершает свою работу.

Запущенная нить T1(F1) запускает нить T2(F2), после чего тем же способом добавляет в формируемый текст свою часть: "Text1, "

После завершения записи соей части текста нить T1(F1) ожидает завершения выполнения нити T2(F2), после чего завершает свою работу.

Запущенная нить T2(F2) должна так же добавить в формируемый текст свою часть:

"Text2.\n "

после чего завершает свою работу.

Для имитации времени записи в текст одной буквы использовать пустой цикл в 1000 итераций.

Применить блокировки чтения/записи, мутексы, присоединение.

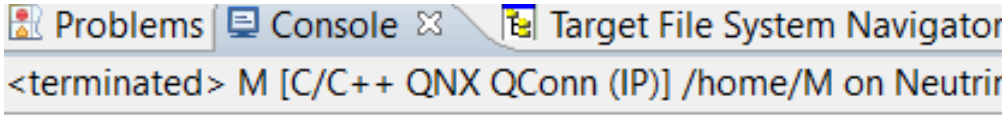
2 Результаты работы

```
#include <iostream>
#include <pthread.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
// --- Константы ---
const int BUF_SIZE = 200;  // размер общего буфера для текста
// --- Глобальные переменные ---
char Text_Buf[BUF_SIZE];       // общий буфер, в который пишут все три нити
pthread_mutex_t buf_mutex;     // мьютекс для синхронизации доступа к буферу
pthread_barrier_t barrier;     // барьер для синхронизации main и T1
// --- Общий флаг для синхронизации ---
int sync_flag = 0;  // 0 - работает main, 1 - работает T1, 2 - работает T2
// --- Аргументы для нитей ---
struct thread_Arg {
   char* Text_Buf;   // указатель на общий буфер
};
// --- Прототипы функций нитей ---
void* F1(void* args);   // функция для нити T1
void* F2(void* args);   // функция для нити T2
void AddSymbol_In_TextBuf(char* TextBuf, char Symbol); // вспомогательная функция записи символа
// --- Функция main выступает как нить M ---
int main(void) {
   pthread_t T1_ID;   // идентификатор нити T1
   // инициализация мьютекса
   pthread_mutex_init(&buf_mutex, NULL);
   // инициализация барьера на 2 участника (main и T1)
   pthread_barrier_init(&barrier, NULL, 2);
   const char mainText[] = "Text0, ";
   std::cout << "Main: старт" << std::endl;
   // формируем аргументы для нити T1
   struct thread_Arg arg;
```

```
    arg.Text_Buf = Text_Buf;
    // запускаем нить T1
    pthread_create(&T1_ID, NULL, &F1, &arg);
    //ожидание флага
    while(sync_flag != 0) sleep(2);
    // Main пишет свой текст "Text0, "
    for (unsigned int i = 0; i < strlen(mainText); i++) {
        pthread_mutex_lock(&buf_mutex);                 // блокируем доступ к буферу
        AddSymbol_In_TextBuf(Text_Buf, mainText[i]);     // добавляем символ
        pthread_mutex_unlock(&buf_mutex);               // разблокируем
    }
    //установка флага для T1
    sync_flag = 1;
    std::cout << "Main: запись выполнил" << std::endl;
    // ждём у барьера (ожидаем, пока T1 закончит свою работу и вызовет barrier)
    pthread_barrier_wait(&barrier);
    // выводим итоговый результат
    std::cout << "Main: итоговый текст:\n" << Text_Buf << std::endl;
    std::cout << "Main: завершает работу" << std::endl;
    // уничтожаем примитивы синхронизации
    pthread_mutex_destroy(&buf_mutex);
    pthread_barrier_destroy(&barrier);
    return EXIT_SUCCESS;
}
// --- Нить T1 ---
void* F1(void* args) {
    struct thread_Arg* arg = (struct thread_Arg*)args;
    pthread_t T2_ID;   // идентификатор нити T2
    std::cout << "T1: старт" << std::endl;
    // T1 запускает нить T2
    pthread_create(&T2_ID, NULL, &F2, arg);
    //ожидание флага
    while(sync_flag != 1) sleep(1);
```

```
    const char text1[] = "Text1, ";
    // T1 пишет свою часть текста "Text1, "
    for (unsigned int i = 0; i < strlen(text1); i++) {
        pthread_mutex_lock(&buf_mutex);                 // блокируем буфер
        AddSymbol_In_TextBuf(arg->Text_Buf, text1[i]);   // пишем символ
        pthread_mutex_unlock(&buf_mutex);               // разблокируем
    }
    //установка флага для T2
    sync_flag = 2;
    // ждём завершения работы нити T2
    pthread_join(T2_ID, NULL);
    std::cout << "T1: запись выполнил и завершает работу" << std::endl;
    // встаём на барьер (ожидаем main)
    pthread_barrier_wait(&barrier);
    return NULL;
}
// --- Нить T2 ---
void* F2(void* args) {
    struct thread_Arg* arg = (struct thread_Arg*)args;
    std::cout << "T2: старт" << std::endl;
    //ожидание флага
    while(sync_flag != 2) usleep(1000);
    const char text2[] = "Text2.\n";
    // T2 пишет свою часть текста "Text2.\n"
    for (unsigned int i = 0; i < strlen(text2); i++) {
        pthread_mutex_lock(&buf_mutex);                 // блокируем буфер
        AddSymbol_In_TextBuf(arg->Text_Buf, text2[i]);   // добавляем символ
        pthread_mutex_unlock(&buf_mutex);               // разблокируем
    }
    std::cout << "T2: запись выполнил и завершает работу" << std::endl;
    return NULL;
}
// --- Вспомогательная функция добавления символа в конец буфера ---
```

```
void AddSymbol_In_TextBuf(char* TextBuf, char Symbol) {
    static int currentIndex = 0;     // текущая позиция в буфере
    TextBuf[currentIndex++] = Symbol; // записываем символ
    TextBuf[currentIndex] = '\0';     // завершение строки
    for (volatile int k = 0; k < 1000; k++);// имитация времени записи символа
}
```




```
Main: старт
T1: старт
Main: запись выполнил
T2: старт
T2: запись выполнил и завершает работу
T1: запись выполнил и завершает работу
Main: итоговый текст:
Text0, Text1, Text2.

Main: завершает работу
```